ASSIGNMENT BRIEF: DATA ANALYST INTERN

PROBLEM SPACE

Datasets:

- File 1: payments.csv (Payment data)
- File 2: customer_orders.csv (Order data)

Tasks Summary:

Use SQL queries to address the following:

(ALL OUTPUT ARE IN GIT HUB OUTPUT FILE)

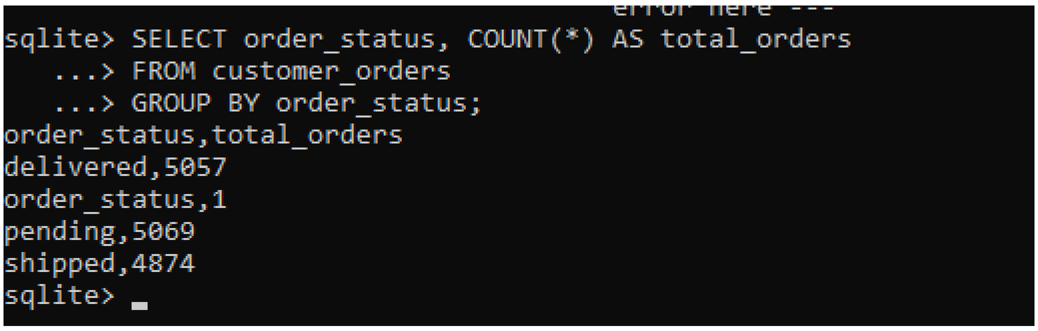
1. **Order and Sales Analysis:**
   - Analyze order status and sales data to provide insights into order fulfillment and revenue trends. Identify key metrics and trends related to order status and sales.

For **Order and Sales Analysis**, we'll write SQL queries that help answer these key questions:

---

1. Count of Orders by Status

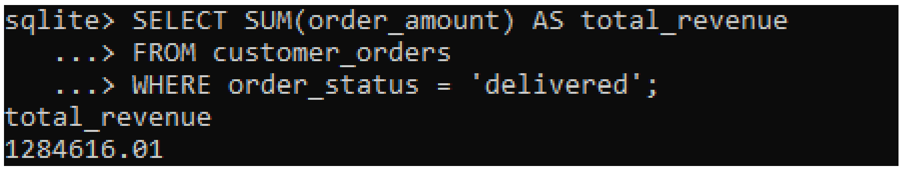
This shows how many orders are completed, pending, etc.



```
                                       error here ---
sqlite> SELECT order_status, COUNT(*) AS total_orders
   ...> FROM customer_orders
   ...> GROUP BY order_status;
order_status,total_orders
delivered,5057
order_status,1
pending,5069
shipped,4874
sqlite> _
```

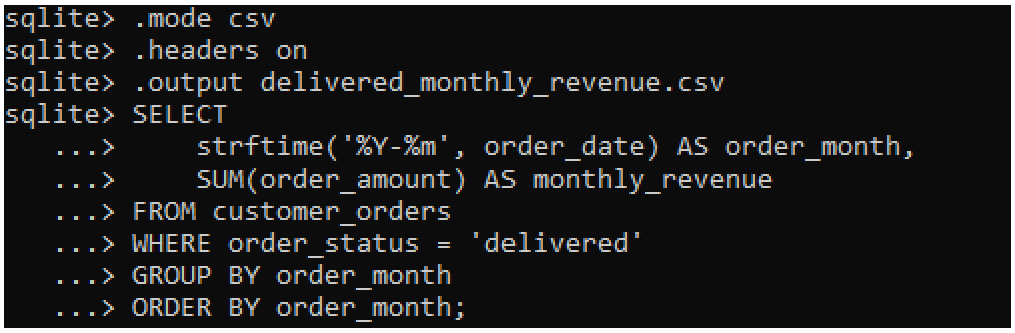
2. Total Revenue from Completed Orders

Only include orders that were marked as "completed".



```
sqlite> SELECT SUM(order_amount) AS total_revenue
   ...> FROM customer_orders
   ...> WHERE order_status = 'delivered';
total_revenue
1284616.01
```

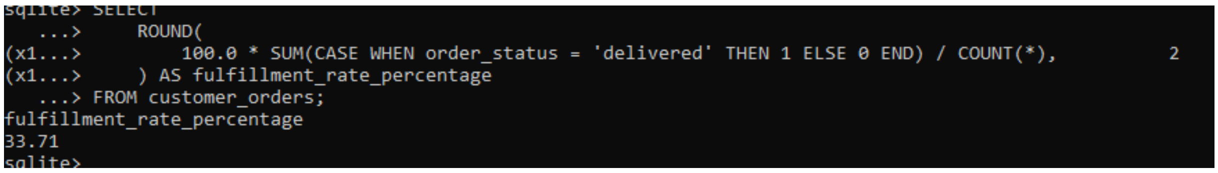
3. **Monthly Revenue Trend**



```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output delivered_monthly_revenue.csv
sqlite> SELECT
   ...>     strftime('%Y-%m', order_date) AS order_month,
   ...>     SUM(order_amount) AS monthly_revenue
   ...> FROM customer_orders
   ...> WHERE order_status = 'delivered'
   ...> GROUP BY order_month
   ...> ORDER BY order_month;
```

repeat_customer.csv

4. **Order Fulfillment Rate**



```
sqlite> SELECT
   ...>     ROUND(
(x1...>         100.0 * SUM(CASE WHEN order_status = 'delivered' THEN 1 ELSE 0 END) / COUNT(*),        2
(x1...>     ) AS fulfillment_rate_percentage
   ...> FROM customer_orders;
fulfillment_rate_percentage
33.71
sqlite>
```

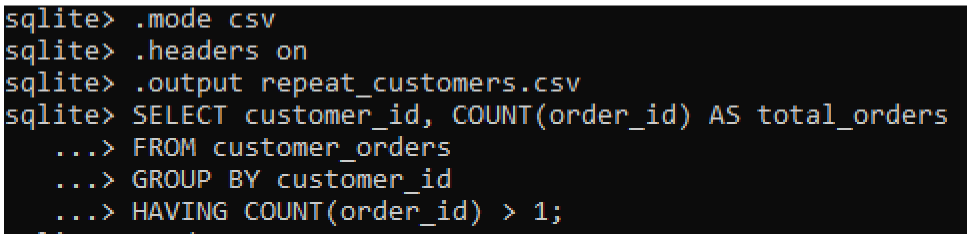
These queries provide a comprehensive overview of order fulfillment and sales trends.

2. **Customer Analysis:**

   - Explore customer ordering behavior to identify patterns such as repeat ordering, customer segmentation, and trends over time.

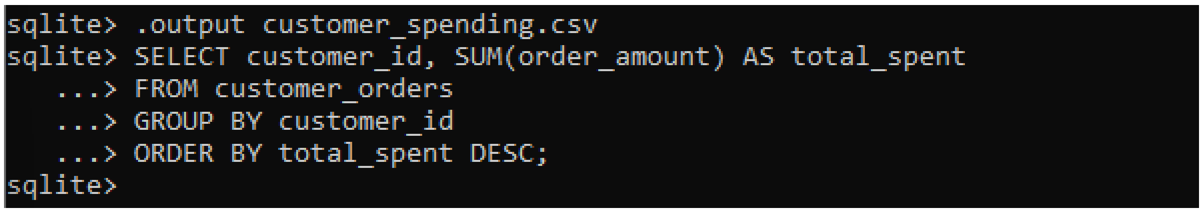
To analyze **customer ordering behavior**, you can write SQL queries for several sub-tasks.

1. Identify Repeat Customers



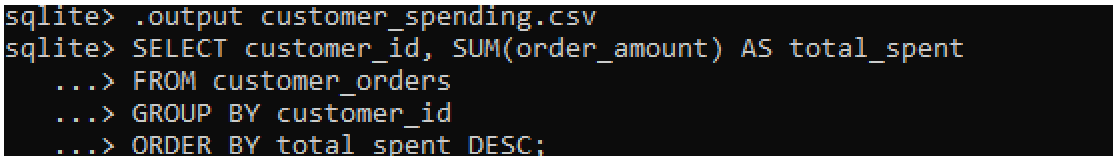
```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output repeat_customers.csv
sqlite> SELECT customer_id, COUNT(order_id) AS total_orders
   ...> FROM customer_orders
   ...> GROUP BY customer_id
   ...> HAVING COUNT(order_id) > 1;
```

2.Total Spending by Each Customer



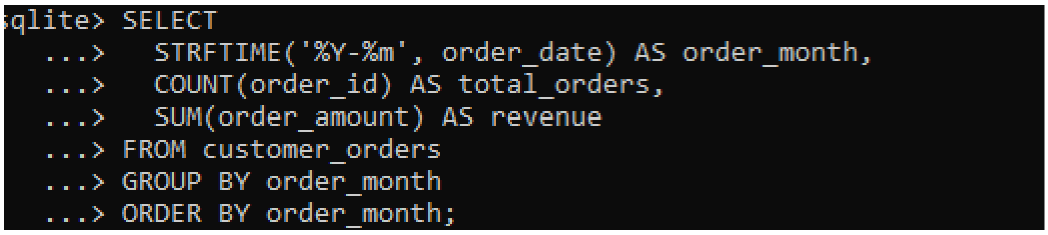
```
sqlite> .output customer_spending.csv
sqlite> SELECT customer_id, SUM(order_amount) AS total_spent
   ...> FROM customer_orders
   ...> GROUP BY customer_id
   ...> ORDER BY total_spent DESC;
sqlite>
```

3. For Total Spending by Customer



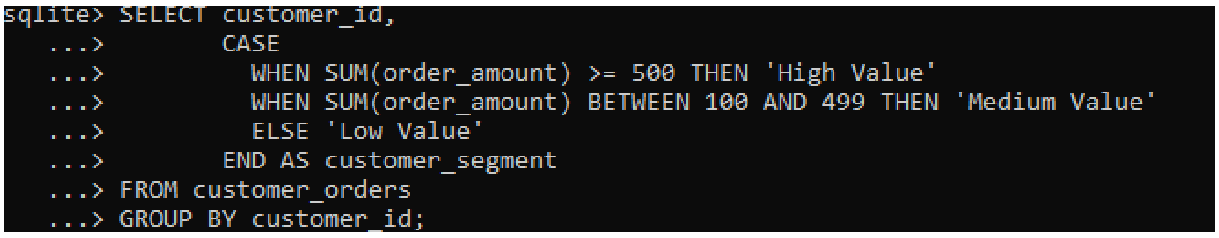
```
sqlite> .output customer_spending.csv
sqlite> SELECT customer_id, SUM(order_amount) AS total_spent
   ...> FROM customer_orders
   ...> GROUP BY customer_id
   ...> ORDER BY total_spent DESC;
```

4. For Monthly Orders and Revenue



```
qlite> SELECT
  ...>   STRFTIME('%Y-%m', order_date) AS order_month,
  ...>   COUNT(order_id) AS total_orders,
  ...>   SUM(order_amount) AS revenue
  ...> FROM customer_orders
  ...> GROUP BY order_month
  ...> ORDER BY order_month;
```

5. For Customer Segmentation

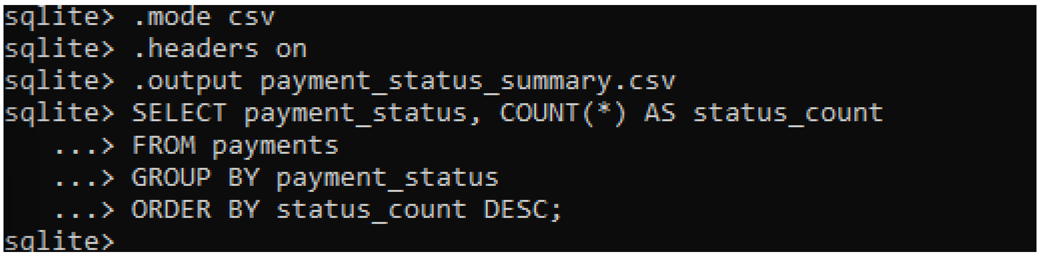


```
sqlite> SELECT customer_id,
   ...>        CASE
   ...>          WHEN SUM(order_amount) >= 500 THEN 'High Value'
   ...>          WHEN SUM(order_amount) BETWEEN 100 AND 499 THEN 'Medium Value'
   ...>          ELSE 'Low Value'
   ...>        END AS customer_segment
   ...> FROM customer_orders
   ...> GROUP BY customer_id;
```

3. **Payment Status Analysis:**

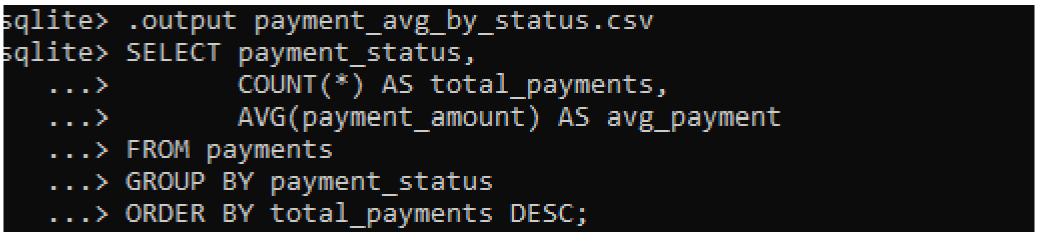
- Investigate payment status data to identify any potential issues or trends related to payment success and failure.

1. We can Export all payment status summary



```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output payment_status_summary.csv
sqlite> SELECT payment_status, COUNT(*) AS status_count
   ...> FROM payments
   ...> GROUP BY payment_status
   ...> ORDER BY status_count DESC;
sqlite>
```

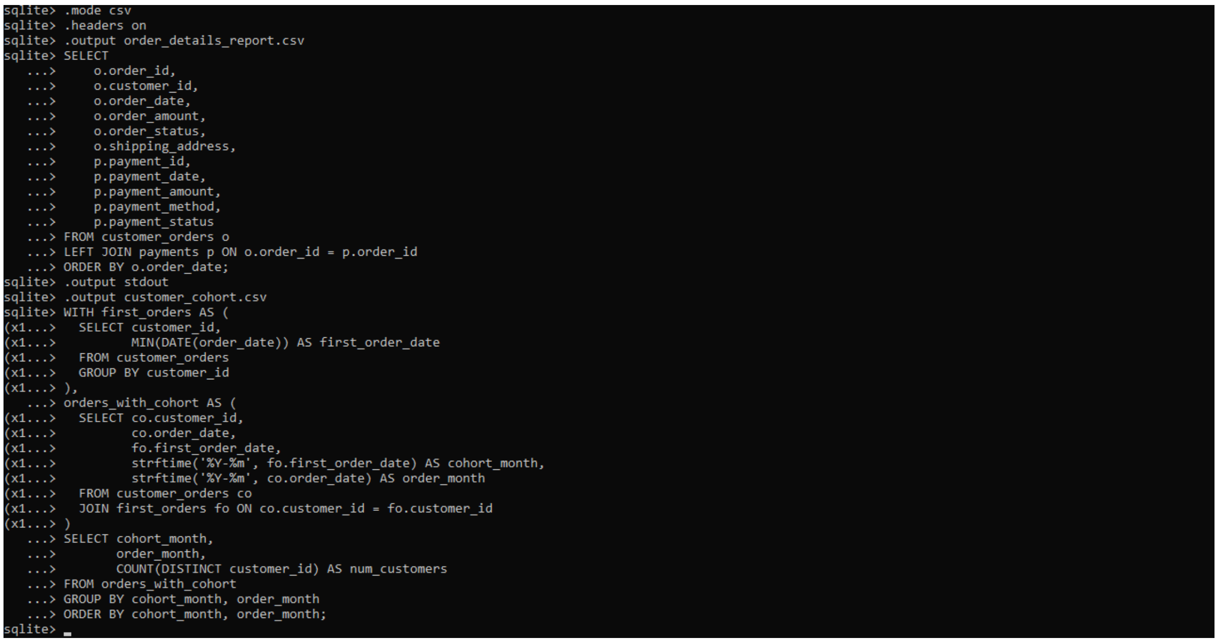
2. We can Export Average Payment By status



```
sqlite> .output payment_avg_by_status.csv
sqlite> SELECT payment_status,
   ...>        COUNT(*) AS total_payments,
   ...>        AVG(payment_amount) AS avg_payment
   ...> FROM payments
   ...> GROUP BY payment_status
   ...> ORDER BY total_payments DESC;
```

4. **Order Details Report:**

   ○ Create a comprehensive report that provides a detailed overview of order information, payment details, and key metrics.



```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output order_details_report.csv
sqlite> SELECT
   ...>     o.order_id,
   ...>     o.customer_id,
   ...>     o.order_date,
   ...>     o.order_amount,
   ...>     o.order_status,
   ...>     o.shipping_address,
   ...>     p.payment_id,
   ...>     p.payment_date,
   ...>     p.payment_amount,
   ...>     p.payment_method,
   ...>     p.payment_status
   ...> FROM customer_orders o
   ...> LEFT JOIN payments p ON o.order_id = p.order_id
   ...> ORDER BY o.order_date;
sqlite> .output stdout
sqlite> .output customer_cohort.csv
sqlite> WITH first_orders AS (
(x1...>   SELECT customer_id,
(x1...>          MIN(DATE(order_date)) AS first_order_date
(x1...>   FROM customer_orders
(x1...>   GROUP BY customer_id
(x1...> ),
   ...> orders_with_cohort AS (
(x1...>   SELECT co.customer_id,
(x1...>          co.order_date,
(x1...>          fo.first_order_date,
(x1...>          strftime('%Y-%m', fo.first_order_date) AS cohort_month,
(x1...>          strftime('%Y-%m', co.order_date) AS order_month
(x1...>   FROM customer_orders co
(x1...>   JOIN first_orders fo ON co.customer_id = fo.customer_id
(x1...> )
   ...> SELECT cohort_month,
   ...>        order_month,
   ...>        COUNT(DISTINCT customer_id) AS num_customers
   ...> FROM orders_with_cohort
   ...> GROUP BY cohort_month, order_month
   ...> ORDER BY cohort_month, order_month;
sqlite> _
```

VISUALIZATION TASK

5. **Customer Retention Analysis:**

   ○ Visualize customer retention by showing how many customers from a specific cohort made repeat purchases in subsequent months.

- Use a suitable BI visualization tool to present your findings.
- Clearly explain how the visualization tracks customer retention.