a. Use SELECT, WHERE, ORDER BY, GROUP BY

```
--- Order by---
SELECT customer_id, customer_city, customer_state
FROM customers
WHERE customer_state = 'SP'
ORDER BY customer_city ASC
LIMIT 5;
-- - Group by ---
SELECT customer_state, COUNT(*) AS num_customers
FROM customers
GROUP BY customer_state
ORDER BY num_customers DESC;
```

b. Use JOINS (INNER, LEFT, RIGHT)

```
-- INNER JOIN: Orders with Customer Details
SELECT o.order_id, c.customer_city
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id
LIMIT 5;
-- LEFT JOIN: All orders, even if customer is missing
SELECT o.order_id, c.customer_city
FROM orders o
LEFT JOIN customers c ON o.customer_id = c.customer_id;
```

c. Write Subqueries

```
SELECT c.customer_id, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING total_orders > 2
LIMIT 5;
```

d. Aggregate Functions (SUM, AVG)

```
-- Total number of order items per order
SELECT order_id, SUM(price) AS total_price
FROM order_items
GROUP BY order_id;
-- Average freight per seller
SELECT seller_id, AVG(freight_value) AS avg_freight
FROM order_items
GROUP BY seller_id
LIMIT 5;
```

e. Create Views for Analysis

```
DROP VIEW IF EXISTS customer_order_count;

CREATE VIEW customer_order_count AS
SELECT customer_id, COUNT(*) AS total_orders
FROM orders
GROUP BY customer_id;
SELECT * FROM customer_order_count LIMIT 5;
```

f. Optimize Queries with Indexes

```
-- Create index on customer_id to speed up joins
CREATE INDEX idx_orders_customer_id ON orders(customer_id);

-- Create index on order_id in order_items
CREATE INDEX idx_order_items_order_id ON order_items(order_id);
```