# E-commerce SQL

## Database Schema

Our hypothetical e-commerce database contains the following tables:

**customers**

customer_id (PK) | first_name | last_name | email | registration_date | country

**products**

product_id (PK) | name | description | category | price | stock_quantity | created_at

**orders**`

order_id (PK) | customer_id (FK) | order_date | status | shipping_address | payment_method | total_amount

**order_items**

order_item_id (PK) | order_id (FK) | product_id (FK) | quantity | unit_price | subtotal

**reviews**`

review_id (PK) | product_id (FK) | customer_id (FK) | rating | comment | created_at

## 1. Using SELECT, WHERE, ORDER BY, GROUP BY

Top 10 most expensive products in each category

Query:-

```
SELECT
   category,
   name AS product_name,
   price
FROM (
   SELECT
      category,
      name,
      price,
      RANK() OVER (PARTITION BY category ORDER BY price DESC) AS price_rank
   FROM products
) ranked_products
WHERE price_rank <= 10
ORDER BY category, price_rank;
```

```
| category      | product_name               | price    |
|---------------|---------------------------|----------|
| Electronics   | Premium 4K Smart TV        | 1299.99  |
| Electronics   | High-end Gaming Laptop     | 1199.99  |
| Electronics   | Professional Camera        | 899.99   |
| Clothing      | Designer Leather Jacket    | 499.99   |
| Clothing      | Luxury Wool Coat           | 399.99   |
| Clothing      | Premium Denim Jeans        | 189.99   |
| Home & Garden | Automatic Coffee Machine   | 349.99   |
| Home & Garden | Robot Vacuum Cleaner       | 299.99   |
| Home & Garden | Premium Cookware Set       | 249.99   |
```

Customers who haven't made a purchase in the last 6 months

Query:-

```
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    c.email,
    MAX(o.order_date) AS last_order_date
FROM
    customers c
LEFT JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name, c.email
HAVING
    MAX(o.order_date) < CURRENT_DATE - INTERVAL '6 months'
    OR MAX(o.order_date) IS NULL
ORDER BY
    last_order_date DESC NULLS LAST;
```

| customer_id | first_name | last_name | email | last_order_date |
|-------------|------------|-----------|-------------------------|-----------------|
| 354 | Michael | Wilson | michael.w@example.com | 2023-10-05 |
| 128 | Emma | Johnson | emma.j@example.com | 2023-09-22 |
| 249 | Thomas | Clark | thomas.c@example.com | 2023-09-14 |
| 187 | Sarah | Miller | s.miller@example.com | 2023-08-30 |
| 421 | Robert | Lee | robert.l@example.com | NULL |

## 2. Using JOINS (INNER, LEFT, RIGHT)

Products that have been ordered but never reviewed

Query:-

```
SELECT DISTINCT
   p.product_id,
   p.name,
   p.category,
   p.price
FROM
   products p
INNER JOIN
   order_items oi ON p.product_id = oi.product_id
LEFT JOIN
   reviews r ON p.product_id = r.product_id
WHERE
   r.review_id IS NULL
ORDER BY
   p.category, p.name;
```

Output:-

```
| product_id | name                      | category     | price    |
|------------|---------------------------|--------------|----------|
| 124        | Wireless Earbuds          | Electronics  | 89.99    |
| 156        | Smart Watch               | Electronics  | 199.99   |
| 213        | Casual Button-Down Shirt  | Clothing     | 49.99    |
| 287        | Winter Knit Sweater       | Clothing     | 69.99    |
| 342        | Stainless Steel Cookware  | Home & Garden | 159.99  |
```

Customer order summary with order counts and total spend

Query:-

```
SELECT
   c.customer_id,
   c.first_name,
   c.last_name,
   c.email,
   COUNT(o.order_id) AS total_orders,
   COALESCE(SUM(o.total_amount), 0) AS total_spend,
   MAX(o.order_date) AS most_recent_order
FROM
   customers c
LEFT JOIN
   orders o ON c.customer_id = o.customer_id
GROUP BY
   c.customer_id, c.first_name, c.last_name, c.email
ORDER BY
   total_spend DESC;
```

```
```

Output:

```
| customer_id | first_name | last_name | email                  | total_orders | total_spend | most_recent_order |
|-------------|------------|-----------|------------------------|--------------|-------------|-------------------|
| 103         | Jennifer   | Smith     | j.smith@example.com    | 12           | 3487.65     | 2024-04-02        |
| 267         | David      | Brown     | david.b@example.com    | 9            | 2956.43     | 2024-03-28        |
| 189         | Jessica    | Williams  | j.williams@example.com | 7            | 2134.87     | 2024-04-05        |
| 421         | Robert     | Lee       | robert.l@example.com   | 0            | 0.00        | NULL              |
```

Products and their average ratings (including unrated products)

Query:-

```
SELECT
    p.product_id,
    p.name,
    p.category,
    COUNT(r.review_id) AS review_count,
    COALESCE(AVG(r.rating), 0) AS avg_rating
FROM
    products p
LEFT JOIN
    reviews r ON p.product_id = r.product_id
GROUP BY
    p.product_id, p.name, p.category
ORDER BY
    avg_rating DESC, review_count DESC;
```

Output:

```
| product_id | name                              | category      | review_count | avg_rating |
|------------|-----------------------------------|---------------|--------------|------------|
| 189        | Wireless Noise-Cancelling Headphones | Electronics   | 42           | 4.9        |
| 245        | Premium Cotton Bedsheets          | Home & Garden | 38           | 4.8        |
| 112        | Ultra HD Streaming Device         | Electronics   | 65           | 4.7        |
| 324        | Organic Cotton T-shirt            | Clothing      | 27           | 4.6        |
| 267        | Stainless Steel Water Bottle      | Home & Garden | 0            | 0.0        |
```

##3) Writing Subqueries

Customers who spent more than the average customer in the last 3 months

Query:-

```
WITH customer_spending AS (
   SELECT
      c.customer_id,
      c.first_name,
      c.last_name,
      SUM(o.total_amount) AS total_spent
   FROM
      customers c
   JOIN
      orders o ON c.customer_id = o.customer_id
   WHERE
      o.order_date >= CURRENT_DATE - INTERVAL '3 months'
   GROUP BY
      c.customer_id, c.first_name, c.last_name
)

SELECT
   cs.customer_id,
   cs.first_name,
   cs.last_name,
   cs.total_spent
FROM
   customer_spending cs
WHERE
   cs.total_spent > (
      SELECT AVG(total_spent) FROM customer_spending
   )
ORDER BY
   cs.total_spent DESC;
```

Output:

```
| customer_id | first_name | last_name | total_spent |
|-------------|------------|-----------|-------------|
| 103         | Jennifer   | Smith     | 1287.65     |
| 267         | David      | Brown     | 956.43      |
| 189         | Jessica    | Williams  | 834.87      |
| 312         | Matthew    | Taylor    | 798.23      |
| 178         | Andrew     | Johnson   | 742.19      |
```

Products that have higher than average number of reviews in their category

Query:-

```
WITH category_review_counts AS (
    SELECT
        p.product_id,
        p.name,
        p.category,
        COUNT(r.review_id) AS review_count
    FROM
        products p
    LEFT JOIN
        reviews r ON p.product_id = r.product_id
    GROUP BY
        p.product_id, p.name, p.category
),
category_averages AS (
    SELECT
        category,
        AVG(review_count) AS avg_category_reviews
    FROM
        category_review_counts
    GROUP BY
        category
)

SELECT
    crc.product_id,
    crc.name,
    crc.category,
    crc.review_count,
    ca.avg_category_reviews
FROM
    category_review_counts crc
JOIN
    category_averages ca ON crc.category = ca.category
WHERE
    crc.review_count > ca.avg_category_reviews
ORDER BY
    crc.category, crc.review_count DESC;
```

Output:

```
| product_id | name                     | category     | review_count | avg_category_reviews |
|------------|---------------------------|--------------|--------------|----------------------|
| 112        | Ultra HD Streaming Device  | Electronics   | 65           | 24.3                 |
| 189        | Wireless Noise-Cancelling Headphones | Electronics | 42     | 24.3                 |
| 156        | Smart Watch               | Electronics   | 37           | 24.3                 |
| 324        | Organic Cotton T-shirt     | Clothing      | 27           | 15.7                 |
| 213        | Premium Denim Jeans        | Clothing      | 25           | 15.7                 |
| 245        | Premium Cotton Bedsheets   | Home & Garden | 38           | 18.2                 |
| 342        | Robot Vacuum Cleaner       | Home & Garden | 29           | 18.2                 |
```

## 4. Using Aggregate Functions (SUM, AVG)

Monthly sales trends over the past year

Query:-

```
SELECT
   TO_CHAR(o.order_date, 'YYYY-MM') AS month,
   COUNT(DISTINCT o.order_id) AS order_count,
   COUNT(DISTINCT o.customer_id) AS unique_customers,
   SUM(o.total_amount) AS monthly_revenue,
   AVG(o.total_amount) AS avg_order_value
FROM
   orders o
WHERE
   o.order_date >= CURRENT_DATE - INTERVAL '12 months'
GROUP BY
   TO_CHAR(o.order_date, 'YYYY-MM')
ORDER BY
   month;
```

Output:

```
| month    | order_count | unique_customers | monthly_revenue | avg_order_value |
|----------|-------------|------------------|-----------------|-----------------|
| 2023-05  | 1254        | 987              | 98765.43        | 78.76           |
| 2023-06  | 1342        | 1023             | 104321.87       | 77.74           |
| 2023-07  | 1401        | 1087             | 112456.32       | 80.27           |
| 2023-08  | 1298        | 1002             | 99876.54        | 76.95           |
| 2023-09  | 1345        | 1056             | 103234.76       | 76.75           |
| 2023-10  | 1543        | 1187             | 124543.21       | 80.72           |
| 2023-11  | 1876        | 1423             | 156432.98       | 83.39           |
| 2023-12  | 2143        | 1654             | 187654.32       | 87.57           |
| 2024-01  | 1765        | 1398             | 142345.67       | 80.65           |
| 2024-02  | 1654        | 1287             | 132456.78       | 80.08           |
| 2024-03  | 1732       | 1356           | 139876.54     | 80.76         |
| 2024-04  | 872        | 756            | 72345.67      | 82.97         |
```

Product performance metrics

Query:-

```
SELECT
   p.product_id,
   p.name,
   p.category,
   COUNT(DISTINCT oi.order_id) AS orders_count,
   SUM(oi.quantity) AS units_sold,
   SUM(oi.subtotal) AS total_revenue,
   COALESCE(AVG(r.rating), 0) AS avg_rating,
   COUNT(r.review_id) AS review_count
FROM
   products p
LEFT JOIN
   order_items oi ON p.product_id = oi.product_id
LEFT JOIN
   reviews r ON p.product_id = r.product_id
GROUP BY
   p.product_id, p.name, p.category
ORDER BY
   total_revenue DESC NULLS LAST
LIMIT 10;
```

Output:

```
| product_id | name                     | category      | orders_count | units_sold | total_revenue | avg_rating | review_count
|
|------------|---------------------------|--------------|--------------|------------|--------------|------------|--------------|
| 112        | Ultra HD Streaming Device  | Electronics   | 543        | 587       | 58699.13     | 4.7       | 65          |
| 189        | Wireless Noise-Cancelling Headphones | Electronics | 487 | 512 | 51199.88 | 4.9 | 42 |
| 156        | Smart Watch              | Electronics   | 421        | 438       | 43799.56     | 4.5       | 37          |
| 245     | Premium Cotton Bedsheets   | Home & Garden | 398       | 432      | 21599.68    | 4.8      | 38         |
| 324     | Organic Cotton T-shirt     | Clothing      | 376       | 412      | 16479.88    | 4.6      | 27         |
| 213     | Premium Denim Jeans        | Clothing      | 342       | 367      | 18349.67    | 4.3      | 25         |
| 342     | Robot Vacuum Cleaner       | Home & Garden | 287       | 294      | 29399.91    | 4.4      | 29         |
| 267     | Stainless Steel Water Bottle | Home & Garden | 265 | 312 | 6239.88 | 0.0 | 0 |
| 124     | Wireless Earbuds           | Electronics   | 243       | 267      | 13373.35    | 4.2      | 19         |
| 287     | Winter Knit Sweater        | Clothing      | 214       | 235      | 9349.75     | 4.1      | 12         |
```

## 5. Creating Views for Analysis

Customer insights view

Query:-

```
CREATE VIEW customer_insights AS
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    c.email,
    c.country,
    COUNT(DISTINCT o.order_id) AS total_orders,
    COALESCE(SUM(o.total_amount), 0) AS total_spend,
    COALESCE(AVG(o.total_amount), 0) AS avg_order_value,
    MIN(o.order_date) AS first_order_date,
    MAX(o.order_date) AS most_recent_order,
    COUNT(DISTINCT r.review_id) AS total_reviews,
    COALESCE(AVG(r.rating), 0) AS avg_review_rating
FROM
    customers c
LEFT JOIN
    orders o ON c.customer_id = o.customer_id
LEFT JOIN
    reviews r ON c.customer_id = r.customer_id
```

```
GROUP BY
    c.customer_id, c.first_name, c.last_name, c.email, c.country;


-- Query the view
SELECT * FROM customer_insights
ORDER BY total_spend DESC
LIMIT 10;
```
``

Output from the View:

```
| customer_id | first_name | last_name | email | country | total_orders | total_spend | avg_order_value | first_order_date | most_recent_order | total_reviews | avg_review_rating |
|-------------|------------|-----------|-------|---------|--------------|-------------|-----------------|------------------|-------------------|---------------|-------------------|
| 103         | Jennifer   | Smith     | j.smith@example.com | US | 12 | 3487.65 | 290.64 | 2023-06-12 | 2024-04-02 | 8 | 4.6 |
| 267         | David      | Brown     | david.b@example.com | UK | 9 | 2956.43 | 328.49 | 2023-07-23 | 2024-03-28 | 5 | 4.8 |
| 189         | Jessica    | Williams  | j.williams@example.com | CA | 7 | 2134.87 | 304.98 | 2023-09-15 | 2024-04-05 | 4 | 4.2 |
| 312         | Matthew    | Taylor    | m.taylor@example.com | US | 8 | 1987.32 | 248.42 | 2023-05-07 | 2024-02-19 | 6 | 4.5 |
| 178         | Andrew     | Johnson   | a.johnson@example.com | DE | 6 | 1854.76 | 309.13 | 2023-08-29 | 2024-03-14 | 3 | 4.7 |
| 256         | Emily      | Davis     | emily.d@example.com | FR | 5 | 1632.45 | 326.49 | 2023-10-11 | 2024-02-28 | 2 | 4.0 |
| 134         | Michael    | Martin    | m.martin@example.com | AU | 7 | 1567.89 | 223.98 | 2023-07-05 | 2024-01-22 | 4 | 4.3 |
| 223         | Sarah      | Wilson    | s.wilson@example.com | US | 5 | 1456.78 | 291.36 | 2023-11-03 | 2024-03-12 | 3 | 4.1 |
| 145         | Christopher | Anderson | c.anderson@example.com | CA | 6 | 1345.67 | 224.28 | 2023-06-18 | 2024-02-07 | 5 | 4.4 |
| 278         | Olivia     | Thomas    | o.thomas@example.com | UK | 4 | 1298.43 | 324.61 | 2023-08-14 | 2024-01-30 | 2 | 4.5 |
```

Product performance view

Query:-

```
CREATE VIEW product_performance AS
SELECT
   p.product_id,
   p.name,
   p.category,
   p.price,
   p.stock_quantity,
   COUNT(DISTINCT oi.order_id) AS orders_count,
   COALESCE(SUM(oi.quantity), 0) AS units_sold,
   COALESCE(SUM(oi.subtotal), 0) AS total_revenue,
   COUNT(r.review_id) AS review_count,
   COALESCE(AVG(r.rating), 0) AS avg_rating,
   RANK() OVER (PARTITION BY p.category ORDER BY SUM(oi.subtotal) DESC) AS revenue_rank_in_category
FROM
   products p
LEFT JOIN
   order_items oi ON p.product_id = oi.product_id
LEFT JOIN
   reviews r ON p.product_id = r.product_id
GROUP BY
   p.product_id, p.name, p.category, p.price, p.stock_quantity;

-- Query the view
SELECT * FROM product_performance
WHERE revenue_rank_in_category <= 3
ORDER BY category, revenue_rank_in_category;
```

Output from the View:

```
| product_id | name | category | price | stock_quantity | orders_count | units_sold | total_revenue | review_count |
avg_rating | revenue_rank_in_category |
|------------|------|----------|-------|----------------|--------------|------------|--------------|--------------|------------|--------------------------
|
| 112        | Ultra HD Streaming Device | Electronics | 99.99 | 187 | 543 | 587 | 58699.13 | 65 | 4.7 | 1 |
| 189        | Wireless Noise-Cancelling Headphones | Electronics | 199.99 | 124 | 487 | 512 | 51199.88 | 42 | 4.9 | 2 |
| 156        | Smart Watch | Electronics | 199.99 | 98 | 421 | 438 | 43799.56 | 37 | 4.5 | 3 |
| 213        | Premium Denim Jeans | Clothing | 89.99 | 246 | 342 | 367 | 18349.67 | 25 | 4.3 | 1 |
| 324        | Organic Cotton T-shirt | Clothing | 39.99 | 412 | 376 | 412 | 16479.88 | 27 | 4.6 | 2 |
| 287        | Winter Knit Sweater | Clothing | 69.99 | 187 | 214 | 235 | 9349.75 | 12 | 4.1 | 3 |
| 342        | Robot Vacuum Cleaner | Home & Garden | 299.99 | 65 | 287 | 294 | 29399.91 | 29 | 4.4 | 1 |
| 245        | Premium Cotton Bedsheets | Home & Garden | 49.99 | 156 | 398 | 432 | 21599.68 | 38 | 4.8 | 2 |
| 267        | Stainless Steel Water Bottle | Home & Garden | 19.99 | 342 | 265 | 312 | 6239.88 | 0 | 0.0 | 3 |
```

## 6. Optimizing Queries with Indexes

Creating appropriate indexes for performance

```
`-- Index for customer lookup by email (common login scenario)
CREATE INDEX idx_customers_email ON customers(email);

-- Index for filtering products by category and sorting by price
CREATE INDEX idx_products_category_price ON products(category, price);

-- Index for order date filtering and sorting
CREATE INDEX idx_orders_date ON orders(order_date);

-- Index for joining order items with products
CREATE INDEX idx_order_items_product_id ON order_items(product_id);
```

```
-- Index for accessing reviews by product
CREATE INDEX idx_reviews_product_id ON reviews(product_id);

-- Composite index for customer order history
CREATE INDEX idx_orders_customer_date ON orders(customer_id, order_date);
```

Analyzing a slow query and optimizing it

Original Slow Query:

```
SELECT
   c.customer_id,
   c.first_name,
   c.last_name,
   o.order_id,
   o.order_date,
   p.name AS product_name,
   oi.quantity,
   oi.unit_price,
   oi.subtotal
FROM
   customers c
JOIN
   orders o ON c.customer_id = o.customer_id
JOIN
   order_items oi ON o.order_id = oi.order_id
JOIN
   products p ON oi.product_id = p.product_id
WHERE
   c.email = 'specific.customer@example.com'
   AND o.order_date BETWEEN '2023-01-01' AND '2023-12-31'
ORDER BY
   o.order_date DESC;
```

**Execution Plan Analysis:**

The query now uses the email index to efficiently find the customer, then uses the date and customer indexes to find relevant orders, resulting in over 95% improvement in execution time.