Intermediate SQL

-- Get products with their brand names

```
SELECT
    p.product_id,
    p.product_name,
    b.brand_name
FROM products p
INNER JOIN brands b ON p.brand_id = b.brand_id
LIMIT 10;
```

product_id	product_name	brand_name
1	Silk Blend Blouse	Chic Threads
2	Urban Runner Sneakers	Urban Edge
3	Organic Cotton Jeans	EcoWear
4	Nordic Wool Peacoat	Nordic Style
5	Tokyo Graphic Tee	Tokyo Street

-- Get all customers and their order IDs, if any

```
SELECT
    c.customer_id,
    c.first_name,
    c.email,
    so.order_id
FROM customers c
LEFT JOIN sales_orders so ON c.customer_id = so.customer_id
LIMIT 15;
```

customer_id	first_name	email	order_id
1	Alice	alice.s@example.com	1
1	Alice	alice.s@example.com	3
1	Alice	alice.s@example.com	8
2	Bob	bob.j@example.com	2
2	Bob	bob.j@example.com	6
3	Charlie	charlie.w@example.com	4
4	Diana	diana.b@example.com	5
5	Ethan	ethan.j@example.com	7

-- Find customers who have NOT placed any orders using LEFT JOIN

```
SELECT
    c.customer_id,
    c.first_name,
    c.email
FROM customers c
LEFT JOIN sales_orders so ON c.customer_id = so.customer_id
WHERE so.order_id IS NULL;
```

customer_id	first_name	email
8	Hannah	hannah.d@example.com
9	Ian	ian.r@example.com
10	Julia	julia.m@example.com

-- Get all orders and their customer names, if any - less common

```
SELECT
    c.first_name,
    c.last_name,
    so.order_id,
    so.order_date
FROM customers c
RIGHT JOIN sales_orders so ON c.customer_id = so.customer_id
LIMIT 15;
```

first_name	last_name	order_id	order_date
Alice	Smith	1	2023-06-15 10:30:00
Bob	Johnson	2	2023-06-20 14:00:00
Alice	Smith	3	2023-07-05 09:00:00
Charlie	Williams	4	2023-07-10 11:45:00
Diana	Brown	5	2023-08-01 16:20:00
Bob	Johnson	6	2023-08-15 10:00:00
Ethan	Jones	7	2023-09-05 13:00:00
Alice	Smith	8	2023-10-10 15:00:00

-- Get all customers and all orders, showing matches and non-matches

```
SELECT c.customer_id, c.first_name, so.order_id, so.order_date
FROM customers c
LEFT JOIN sales_orders so ON c.customer_id = so.customer_id
UNION
```

```
SELECT c.customer_id, c.first_name, so.order_id, so.order_date
FROM customers c
RIGHT JOIN sales_orders so ON c.customer_id = so.customer_id;
```

customer_id	first_name	order_id	order_date
1	Alice	1	2023-06-15 10:30:00
1	Alice	3	2023-07-05 09:00:00
1	Alice	8	2023-10-10 15:00:00
2	Bob	2	2023-06-20 14:00:00
2	Bob	6	2023-08-15 10:00:00
3	Charlie	4	2023-07-10 11:45:00
4	Diana	5	2023-08-01 16:20:00
5	Ethan	7	2023-09-05 13:00:00

-- Joining Multiple Tables (Customer -> Order -> OrderItem -> InventoryItem -> Product).

```
SELECT
    c.first_name AS CustomerName,
    so.order_id AS OrderID,
    so.order_date AS OrderDate,
    p.product_name AS ProductName,
    oi.selling_price AS PriceSold,
    oi.discount_amount AS Discount
FROM customers c
JOIN sales_orders so ON c.customer_id = so.customer_id
JOIN order_items oi ON so.order_id = oi.order_id
JOIN inventory_items ii ON oi.inventory_item_id = ii.inventory_item_id
JOIN products p ON ii.product_id = p.product_id
WHERE c.customer_id = 1 -- Example: Filter for a specific customer
ORDER BY so.order_date DESC;
```

CustomerName	OrderID	OrderDate	ProductName	PriceSold	Discount
Alice	8	2023-10-10 15:00:00	Seoul Leather Handbag	299.99	0.00
Alice	3	2023-07-05 09:00:00	Tokyo Graphic Tee	59.99	0.00
Alice	1	2023-06-15 10:30:00	Silk Blend Blouse	149.99	0.00

-- Explicit CROSS JOIN (Example: Combine first 3 brands and first 3 categories)

```
SELECT
        b.brand_name, c.category_name
FROM (SELECT brand_name FROM brands LIMIT 3) b
        CROSS JOIN
                (SELECT category_name FROM categories
                        WHERE parent_category_id IS NOT NULL LIMIT 3) c;
```

brand_name	category_name
Chic Threads	Menswear
Berlin Basics	Menswear
Aussie Vibes	Menswear
Chic Threads	Womenswear
Berlin Basics	Womenswear
Aussie Vibes	Womenswear
Chic Threads	Outerwear
Berlin Basics	Outerwear

-- Get a combined list of customer emails and supplier emails (remove duplicates)

```
SELECT email, "Customer" as source_type FROM customers WHERE email IS NOT NULL
UNION
SELECT contact_email, "Supplier" as source_type FROM suppliers WHERE contact_email
IS NOT NULL;
```

email	source_type
alice.s@example.com	Customer
bob.j@example.com	Customer
charlie.w@example.com	Customer
diana.b@example.com	Customer
ethan.j@example.com	Customer
fiona.g@example.com	Customer
george.m@example.com	Customer
hannah.d@example.com	Customer

-- Get a combined list including duplicates

```
SELECT email, "Customer" as source_type FROM customers WHERE email IS NOT NULL
UNION ALL
SELECT contact_email, "Supplier" as source_type FROM suppliers WHERE contact_email
IS NOT NULL;
```

email	source_type
alice.s@example.com	Customer
bob.j@example.com	Customer
charlie.w@example.com	Customer
diana.b@example.com	Customer
ethan.j@example.com	Customer
fiona.g@example.com	Customer
george.m@example.com	Customer
hannah.d@example.com	Customer

-- Data Cleaning: Standardize condition names

```
SELECT
    inventory_item_id,
    sku,
    CASE
                    WHEN current_condition_id = 1 THEN "New"
        WHEN current_condition_id = 2 THEN "Like New"
        WHEN current_condition_id = 3 THEN "Excellent"
        WHEN current_condition_id = 4 THEN "Good"
        WHEN current_condition_id = 5 THEN "Fair"
        ELSE "Unknown"
    END AS condition_label
FROM inventory_items
LIMIT 10;
```

inventory_item_id	sku	condition_label
1	SKU001	New
2	SKU002	New
3	SKU003	New
4	SKU004	New
5	SKU005	Like New
6	SKU006	New
7	SKU007	New
8	SKU008	New

-- Analysis: Categorize orders by total amount

```
SELECT
    order_id,
    total_amount,
    CASE
        WHEN total_amount < 50 THEN "Small"
        WHEN total_amount < 200 THEN "Medium"
```

```
        WHEN total_amount >= 200 THEN "Large"
        ELSE "Unknown"
    END AS order_size_category
FROM sales_orders
LIMIT 15;
```

order_id	total_amount	order_size_category
1	149.99	Medium
2	89.99	Medium
3	59.99	Medium
4	179.99	Medium
5	399.99	Large
6	49.99	Small
7	249.99	Large
8	299.99	Large

-- Feature Engineering: Flag orders with discounts

```
SELECT
    order_id,
    SUM(discount_amount) AS total_discount,
    CASE
        WHEN SUM(discount_amount) > 0 THEN "Yes"
        ELSE "No"
    END AS has_discount
FROM order_items
GROUP BY order_id
LIMIT 10;
```

order_id	total_discount	has_discount
1	0.00	No
2	0.00	No
3	0.00	No
4	0.00	No
5	0.00	No
6	0.00	No
7	0.00	No
8	0.00	No

-- Conditional Aggregation: Count orders by status per customer

```
SELECT
    customer_id,
    COUNT(CASE WHEN order_status = "Delivered" THEN order_id END) AS
delivered_count,
```

```
    COUNT(CASE WHEN order_status = "Shipped" THEN order_id END) AS shipped_count,
    COUNT(CASE WHEN order_status = "Cancelled" THEN order_id END) AS
cancelled_count,
    COUNT(CASE WHEN order_status = "Returned" THEN order_id END) AS
returned_count
FROM sales_orders
GROUP BY customer_id
LIMIT 10;
```

customer_id	delivered_count	shipped_count	cancelled_count	returned_count
1	3	0	0	0
2	2	0	0	0
3	0	1	0	0
4	0	0	0	0
5	1	0	0	0
6	0	0	1	0
7	1	0	0	0

-- Query the view like a table

```
SELECT *
FROM view_product_details
WHERE brand_name = "Chic Threads";
```

product_id	product_name	brand_name	category_name	purchase_price	material	color	size
1	Silk Blend Blouse	Chic Threads	Womens Tops	85.50	Silk Blend	Cream	M

-- Query the view to find average price per brand

```
SELECT brand_name, AVG(purchase_price) AS avg_price
FROM view_product_details
GROUP BY brand_name
ORDER BY avg_price DESC;
```

brand_name	avg_price
Nordic Style	250.000000
Seoul Select	220.000000
Parisian Flair	180.000000
Urban Edge	120.000000
Aussie Vibes	95.000000
Chic Threads	85.500000
EcoWear	65.000000
Tokyo Street	45.000000

-- Create a simple stored procedure to get orders for a specific customer

```
DELIMITER //

CREATE PROCEDURE GetCustomerOrders (IN cust_id INT)
BEGIN
    SELECT
        order_id,
        order_date,
        total_amount,
        order_status
    FROM sales_orders
    WHERE customer_id = cust_id
    ORDER BY order_date DESC;
END //

DELIMITER ;
```