SQL CASE STUDY

DATA IN MOTION
TINY SHOP SALES

DATA IN MOTION

```sql
-- Create tables and insert data for analysis

DROP TABLE IF EXISTS customers
CREATE TABLE customers (
    customer_id integer PRIMARY KEY,
    first_name varchar(100),
    last_name varchar(100),
    email varchar(100)
)

DROP TABLE IF EXISTS products
CREATE TABLE products (
    product_id integer PRIMARY KEY,
    product_name varchar(100),
    price decimal
)

DROP TABLE IF EXISTS orders
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    customer_id integer,
    order_date date
)

DROP TABLE IF EXISTS order_items
CREATE TABLE order_items (
    order_id integer,
    product_id integer,
    quantity integer
)

INSERT INTO customers (customer_id, first_name, last_name, email) VALUES
(1, 'John', 'Doe', 'johndoe@email.com'),
(2, 'Jane', 'Smith', 'janesmith@email.com'),
(3, 'Bob', 'Johnson', 'bobjohnson@email.com'),
(4, 'Alice', 'Brown', 'alicebrown@email.com'),
```

```
(5, 'Charlie', 'Davis', 'charliedavis@email.com'),
(6, 'Eva', 'Fisher', 'evafisher@email.com'),
(7, 'George', 'Harris', 'georgeharris@email.com'),
(8, 'Ivy', 'Jones', 'ivyjones@email.com'),
(9, 'Kevin', 'Miller', 'kevinmiller@email.com'),
(10, 'Lily', 'Nelson', 'lilynelson@email.com'),
(11, 'Oliver', 'Patterson', 'oliverpatterson@email.com'),
(12, 'Quinn', 'Roberts', 'quinnroberts@email.com'),
(13, 'Sophia', 'Thomas', 'sophiathomas@email.com')

INSERT INTO products (product_id, product_name, price) VALUES
(1, 'Product A', 10.00),
(2, 'Product B', 15.00),
(3, 'Product C', 20.00),
(4, 'Product D', 25.00),
(5, 'Product E', 30.00),
(6, 'Product F', 35.00),
(7, 'Product G', 40.00),
(8, 'Product H', 45.00),
(9, 'Product I', 50.00),
(10, 'Product J', 55.00),
(11, 'Product K', 60.00),
(12, 'Product L', 65.00),
(13, 'Product M', 70.00)

INSERT INTO orders (order_id, customer_id, order_date) VALUES
(1, 1, '2023-05-01'),
(2, 2, '2023-05-02'),
(3, 3, '2023-05-03'),
(4, 1, '2023-05-04'),
(5, 2, '2023-05-05'),
(6, 3, '2023-05-06'),
(7, 4, '2023-05-07'),
(8, 5, '2023-05-08'),
(9, 6, '2023-05-09'),
(10, 7, '2023-05-10'),
(11, 8, '2023-05-11'),
```

```
(12, 9, '2023-05-12'),
(13, 10, '2023-05-13'),
(14, 11, '2023-05-14'),
(15, 12, '2023-05-15'),
(16, 13, '2023-05-16')

INSERT INTO order_items (order_id, product_id, quantity) VALUES
(1, 1, 2),
(1, 2, 1),
(2, 2, 1),
(2, 3, 3),
(3, 1, 1),
(3, 3, 2),
(4, 2, 4),
(4, 3, 1),
(5, 1, 1),
(5, 3, 2),
(6, 2, 3),
(6, 1, 1),
(7, 4, 1),
(7, 5, 2),
(8, 6, 3),
(8, 7, 1),
(9, 8, 2),
(9, 9, 1),
(10, 10, 3),
(10, 11, 2),
(11, 12, 1),
(11, 13, 3),
(12, 4, 2),
(12, 5, 1),
(13, 6, 3),
(13, 7, 2),
(14, 8, 1),
(14, 9, 2),
(15, 10, 3),
(15, 11, 1),
```

```
(16, 12, 2),
(16, 13, 3)


-- Case Study Questions

--1) Which product has the highest price? Only return a single row.

SELECT TOP 1 product_name
FROM products
ORDER BY price DESC
```

Results Messages

	product_name
1	Product M

```
--2) Which customer has made the most orders?

SELECT TOP 1 c.first_name, c.last_name, COUNT(o.order_id) AS order_count
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.first_name, c.last_name
ORDER BY order_count DESC
```

Results Messages

	first_name	last_name	order_count
1	John	Doe	2

```
--3) What's the total revenue per product?

SELECT p.product_name, SUM(p.price * oi.quantity) AS total_revenue
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
```

```
GROUP BY p.product_name
ORDER BY total_revenue DESC
```

Results Messages

	product_name	total_revenue
1	Product M	420
2	Product J	330
3	Product F	210
4	Product L	195
5	Product K	180
6	Product C	160
7	Product I	150
8	Product B	135
9	Product H	135
10	Product G	120
11	Product E	90
12	Product D	75
13	Product A	50

```
--4) Find the day with the highest revenue.

SELECT TOP 1 order_date, SUM(p.price * oi.quantity) AS daily_revenue
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY order_date
ORDER BY daily_revenue DESC
```

Results Messages

	order_date	daily_revenue
1	2023-05-16	340

```
--5) Find the first order (by date) for each customer.

SELECT c.first_name, c.last_name, MIN(o.order_date) AS first_order_date
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.first_name, c.last_name
```

Results Messages

	first_name	last_name	first_order_date
1	Alice	Brown	2023-05-07
2	Charlie	Davis	2023-05-08
3	John	Doe	2023-05-01
4	Eva	Fisher	2023-05-09
5	George	Harris	2023-05-10
6	Bob	Johnson	2023-05-03
7	Ivy	Jones	2023-05-11
8	Kevin	Miller	2023-05-12
9	Lily	Nelson	2023-05-13
10	Oliver	Patterson	2023-05-14
11	Quinn	Roberts	2023-05-15
12	Jane	Smith	2023-05-02
13	Sophia	Thomas	2023-05-16

```
--6) Find the top 3 customers who have ordered the most distinct products

SELECT TOP 3 c.first_name, c.last_name, COUNT(DISTINCT oi.product_id) AS distinct_product_count
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.first_name, c.last_name
ORDER BY distinct_product_count DESC
```

Results Messages

	first_name	last_name	distinct_product_count
1	John	Doe	3
2	Bob	Johnson	3
3	Jane	Smith	3

```
--7) Which product has been bought the least in terms of quantity?

SELECT TOP 1 product_name, SUM(quantity) AS total_quantity_sold
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY product_name
ORDER BY total_quantity_sold ASC;
```

Results Messages

	product_name	total_quantity_sold
1	Product D	3

```
--8) What is the median order total?

WITH OrderedOrders AS (
  SELECT
    o.order_id,
    SUM(p.price * oi.quantity) AS order_total
  FROM
    orders o
  JOIN
    order_items oi ON o.order_id = oi.order_id
  JOIN
    products p ON oi.product_id = p.product_id
  GROUP BY
    o.order_id
```

```
),
OrderedAndNumbered AS (
    SELECT
        order_id,
        order_total,
        ROW_NUMBER() OVER (ORDER BY order_total) AS OrderRank,
        COUNT(*) OVER () AS TotalOrders
    FROM
        OrderedOrders
)

SELECT
    CASE
        WHEN TotalOrders % 2 = 1 THEN
            (SELECT order_total FROM OrderedAndNumbered WHERE OrderRank = (TotalOrders + 1) / 2)
        ELSE
            (SELECT AVG(order_total * 1.0) FROM OrderedAndNumbered WHERE OrderRank IN (TotalOrders / 2,
(TotalOrders / 2) + 1))
    END AS median_order_total
FROM
    OrderedAndNumbered
```

Results Messages

	median_order_total
1	113

```
--9) For each order, determine if it was 'Expensive' (total over 300), 'Affordable' (total over 100), or 'Cheap'.

SELECT o.order_id,
       SUM(p.price * oi.quantity) AS order_total,
       CASE
           WHEN SUM(p.price * oi.quantity) > 300 THEN 'Expensive'
           WHEN SUM(p.price * oi.quantity) > 100 THEN 'Affordable'
           ELSE 'Cheap'
       END AS order_category
```

```
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY o.order_id
```

Results Messages

	order_id	order_total	order_category
1	1	35	Cheap
2	2	75	Cheap
3	3	50	Cheap
4	4	80	Cheap
5	5	50	Cheap
6	6	55	Cheap
7	7	85	Cheap
8	8	145	Affordable
9	9	140	Affordable
10	10	285	Affordable
11	11	275	Affordable
12	12	80	Cheap
13	13	185	Affordable
14	14	145	Affordable
15	15	225	Affordable
16	16	340	Expensive

```
--10) Find customers who have ordered the product with the highest price.

SELECT c.first_name, c.last_name, p.product_name, p.price AS product_price
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 p.price = (SELECT MAX(price) FROM products)
```

Results Messages

	first_name	last_name	product_name	product_price
1	Ivy	Jones	Product M	70
2	Sophia	Thomas	Product M	70