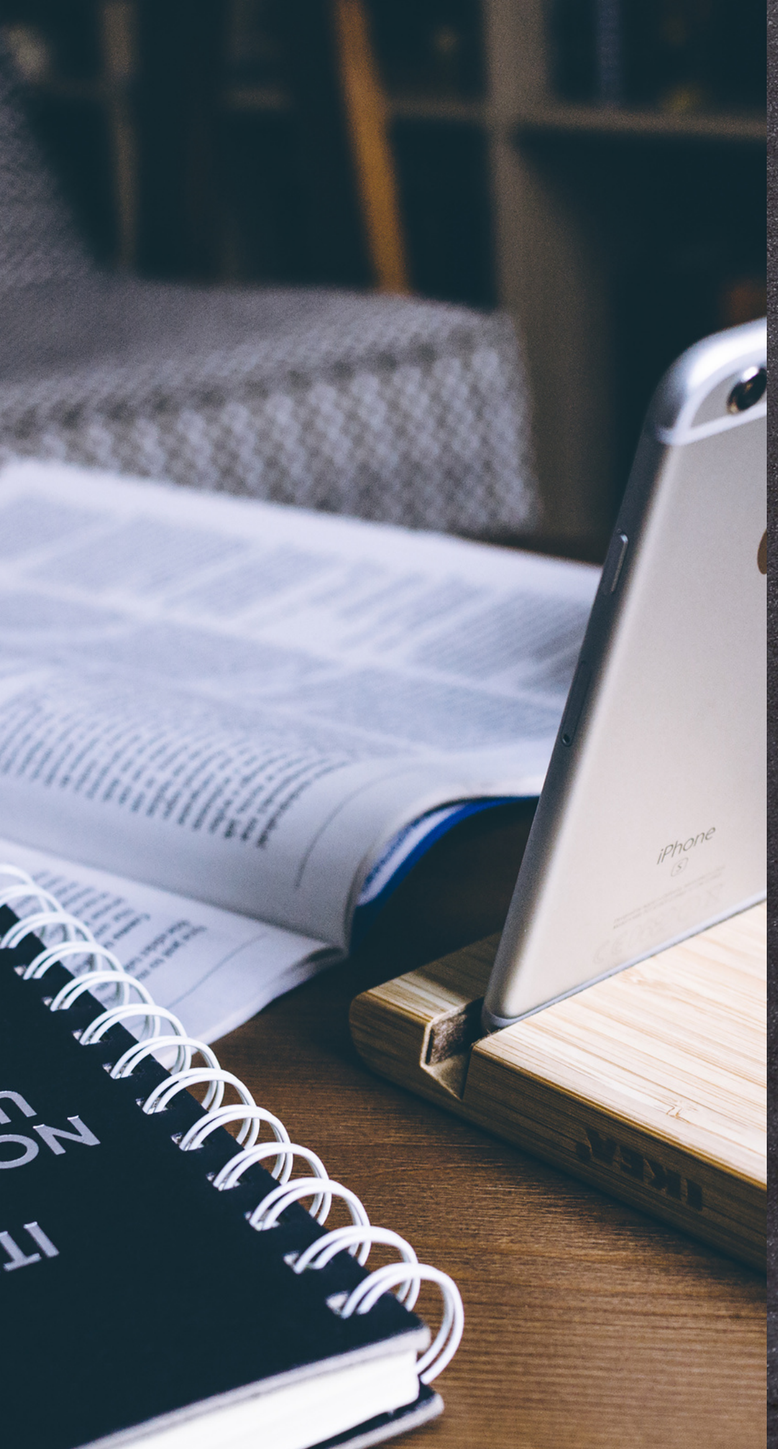
SQL CASE STUDY

DATA IN MOTION
TINY SHOP SALES

DATA IN MOTION

By Sanjay Divate



Introduction

- Tiny Shop is an online retail store with four interconnected tables: customers, order_items, orders, and products.
- These tables store customer information, item details, order information, and product details, respectively.
- By utilizing this comprehensive database structure, Tiny Shop effectively manages customers, tracks inventory, processes orders, and analyzes sales data, leading to excellent customer service and optimized operations for success in the online retail market.

Tables

Customers

customer_id	first_name	last_name	email
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.c...
12	Quinn	Roberts	quinnroberts@email.com
13	Sophia	Thomas	sophiathomas@email.com

Orders >>>

order_id	customer_id	order_date
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

Order Items >>>

order_id	product_id	quantity
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

Products

product_id	product_name	price
1	Product A	10
2	Product B	15
3	Product C	20
4	Product D	25
5	Product E	30
6	Product F	35
7	Product G	40
8	Product H	45
9	Product I	50
10	Product J	55
11	Product K	60
12	Product L	65
13	Product M	70

Questions

1. Which product has the highest price? Only return a single row.
2. Which customer has made the most orders?
3. What’s the total revenue per product?
4. Find the day with the highest revenue.
5. Find the first order (by date) for each customer.
6. Find the top 3 customers who have ordered the most distinct products
7. Which product has been bought the least in terms of quantity?
8. What is the median order total?
9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.
10. Find customers who have ordered the product with the highest price.

QUESTION 1 :

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

Alternative 1:

```sql
SELECT product_id,
       product_name,
       price
FROM products
ORDER BY price DESC
LIMIT 1;
```

Alternative 2:

```sql
SELECT product_id,
       product_name,
       price
FROM products
WHERE PRICE = (SELECT MAX(price) FROM products);
```

product_id	product_name	price
13	Product M	70
NULL	NULL	NULL

Result

QUESTION 2 :

Which customer has made the most orders?

Query:

```sql
SELECT c.customer_id,
       c.first_name,
       c.last_name,
       COUNT(o.order_id) AS 'total orders'
FROM customers c
JOIN orders o
USING(customer_id)
GROUP BY o.customer_id
ORDER BY COUNT(o.order_id) DESC;
```

Result

customer_id	first_name	last_name	total orders
1	John	Doe	2
2	Jane	Smith	2
3	Bob	Johnson	2
4	Alice	Brown	1
5	Charlie	Davis	1
6	Eva	Fisher	1
7	George	Harris	1
8	Ivy	Jones	1
9	Kevin	Miller	1
10	Lily	Nelson	1
11	Oliver	Patterson	1
12	Quinn	Roberts	1
13	Sophia	Thomas	1

QUESTION 3 :

What’s the total revenue per product?

Query:

```sql
SELECT p.product_id,
       p.product_name,
       SUM(p.price*o.quantity) AS 'Total Revenue'
FROM products p
JOIN order_items o
USING(product_id)
GROUP BY o.product_id
ORDER BY SUM(p.price*o.quantity) DESC;
```

Result

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

QUESTION 4 :

Find the day with the highest revenue.

Query:

```
SELECT o.order_date AS DATE,
       DAYNAME(o.order_date)AS Day_name,
       SUM(p.price*ot.quantity) AS 'Total Revenue'
FROM orders o
JOIN order_items ot
USING(order_id)
JOIN products p
USING(product_id)
GROUP BY o.order_date
ORDER BY  SUM(p.price*ot.quantity) DESC;
```

Result

DATE	Day_name	Total Revenue
2023-05-16	Tuesday	340
2023-05-10	Wednesday	285
2023-05-11	Thursday	275
2023-05-15	Monday	225
2023-05-13	Saturday	185
2023-05-08	Monday	145
2023-05-14	Sunday	145
2023-05-09	Tuesday	140
2023-05-07	Sunday	85
2023-05-04	Thursday	80
2023-05-12	Friday	80
2023-05-02	Tuesday	75
2023-05-06	Saturday	55
2023-05-03	Wednesday	50
2023-05-05	Friday	50
2023-05-01	Monday	35

QUESTION 5 :

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

Alternative 1:

```sql
SELECT customer_id,
       first_name,
       last_name,
       order_date
FROM (
  SELECT c.customer_id, c.first_name,c.last_name, o.order_date,
    RANK() OVER (PARTITION BY c.customer_id ORDER BY o.order_date) AS order_rank
  FROM customers c
  JOIN orders o
  USING(customer_id)
) AS ranked_orders
WHERE order_rank = 1;
```

Alternative 2:

```sql
SELECT c.customer_id,
       c.first_name,
       c.last_name,
       MIN(o.order_date) AS First_order_date
FROM customers c
JOIN orders o
USING(customer_id)
GROUP BY 1;
```

Result

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

QUESTION 6 :

Find the top 3 customers who have ordered the most distinct products

Query:

```sql
SELECT c.customer_id,
       CONCAT(c.first_name,' ', c.last_name) as 'Full Name',
       COUNT(DISTINCT ot.product_id) AS distinct_product_count
FROM customers c
JOIN orders o
USING(customer_id)
JOIN order_items ot
USING(order_id)
GROUP BY 1,2
ORDER BY distinct_product_count DESC
LIMIT 3;
```

Result

customer_id	Full Name	distinct_product_count
1	John Doe	3
2	Jane Smith	3
3	Bob Johnson	3

QUESTION 7 :

Which product has been bought the least in terms of quantity?

Query:

```sql
SELECT p.product_id,
       p.product_name,
       SUM(ot.quantity) as "Total Quantity"
FROM products p
JOIN order_items ot
USING(product_id)
GROUP BY 1
ORDER BY SUM(ot.quantity);
```

Result

	product_id	product_name	Total Quantity
▶	4	Product D	3
	5	Product E	3
	7	Product G	3
	8	Product H	3
	9	Product I	3
	11	Product K	3
	12	Product L	3
	1	Product A	5
	6	Product F	6
	10	Product J	6
	13	Product M	6
	3	Product C	8
	2	Product B	9

QUESTION 8 :

What is the median order total ?

Query:

```sql
SELECT round(avg(total_order),2) AS median
FROM(
SELECT ot.order_id,sum(p.price*ot.quantity) AS total_order
FROM products p
JOIN order_items ot
USING(product_id)
GROUP BY 1
ORDER BY total_order DESC
)
as Total;
```

Result

median
140.63

QUESTION 9 :

For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.

Query:

```
SELECT o.order_id,
case
    WHEN SUM(ot.quantity*p.price)>300 THEN "Expensive"
    WHEN SUM(ot.quantity*p.price)>100 THEN "Affordable"
    ELSE "cheap"
END  AS "Budget Category"
FROM order_items ot
JOIN orders o
USING(order_id)
JOIN products p
USING(product_id)
GROUP BY 1;
```

Result

order_id	Budget Category
1	cheap
2	cheap
3	cheap
4	cheap
5	cheap
6	cheap
7	cheap
8	Affordable
9	Affordable
10	Affordable
11	Affordable
12	cheap
13	Affordable
14	Affordable
15	Affordable
16	Expensive

QUESTION 10 :

Find customers who have ordered the product with the highest price.

Query:

```sql
SELECT c.customer_id,
       CONCAT(c.first_name,' ',c.last_name) AS Full_name,
       MAX(p.price) AS Highest_price
FROM customers c
JOIN orders o
USING(customer_id)
JOIN order_items ot
USING(order_id)
JOIN products p
USING(product_id)
WHERE p.price = (SELECT max(price) FROM products)
GROUP BY 1,2;
```

Result

customer_id	Full_name	Highest_price
8	Ivy Jones	70
13	Sophia Thomas	70


THANK YOU!