

SQL ASSIGNMENT 3

Dataset: Sales Information

You have been given a dataset containing information about sales transactions. The dataset includes the following columns:

- `order_id` (integer): Unique identifier for each order.
- `customer_id` (integer): Unique identifier for each customer.
- `product_id` (integer): Unique identifier for each product.
- `product_name` (string): Name of the product.
- `quantity` (integer): The quantity of the product sold.
- `unit_price` (decimal): The unit price of the product.
- `order_date` (date): The date when the order was placed.

Table Structure:

Create a table named `sales` with the following structure:

```
CREATE OR REPLACE TABLE SALES
(
  ORDER_ID INT PRIMARY KEY,
  CUSTOMER_ID INT,
  PRODUCT_ID INT,
  PRODUCT_NAME VARCHAR(50),
  QUANTITY INT,
  UNIT_PRICE DECIMAL(10, 2),
  ORDER_DATE DATE
);
```

Insert Data:

Insert the following sample data into the `sales` table:

```
INSERT INTO SALES (ORDER_ID, CUSTOMER_ID, PRODUCT_ID, PRODUCT_NAME, QUANTITY, UNIT_PRICE, ORDER_DATE)
VALUES
  (1, 101, 1, 'Widget A', 5, 10.00, '2023-01-15'),
  (2, 102, 2, 'Widget B', 2, 12.50, '2023-01-16'),
  (3, 103, 1, 'Widget A', 3, 10.00, '2023-01-16'),
```

```
(4, 104, 3, 'Widget C', 1, 15.75, '2023-01-17'),
(5, 105, 2, 'Widget B', 4, 12.50, '2023-01-17'),
(6, 106, 1, 'Widget A', 2, 10.00, '2023-01-18'),
(7, 107, 4, 'Widget D', 3, 20.00, '2023-01-18'),
(8, 108, 2, 'Widget B', 5, 12.50, '2023-01-19'),
(9, 109, 1, 'Widget A', 1, 10.00, '2023-01-19'),
(10, 101, 3, 'Widget C', 2, 15.75, '2023-01-20');
```

Instructions:

Write SQL queries to answer the following questions using the sales table:

1. Retrieve the total sales quantity and revenue for each product.

```
SELECT PRODUCT_NAME,
SUM(QUANTITY) AS TOT_SALES_QUANTITY,
SUM(UNIT_PRICE * QUANTITY) AS REVENUE
FROM EMP_DATABASE.PUBLIC.SALES
GROUP BY 1
ORDER BY 1;
```

```
34      -- 1. Retrieve the total sales quantity and revenue for each product.
35
36      SELECT PRODUCT_NAME,
37      SUM(QUANTITY) AS TOT_SALES_QUANTITY,
38      SUM(UNIT_PRICE * QUANTITY) AS REVENUE
39      FROM EMP_DATABASE.PUBLIC.SALES
40      GROUP BY 1
41      ORDER BY 1;
```

Results

Chart

	PRODUCT_NAME	TOT_SALES_QUANTITY	REVENUE
1	Widget A	11	110.00
2	Widget B	11	137.50
3	Widget C	3	47.25
4	Widget D	3	60.00

2. Find the total revenue for each customer.

```
SELECT CUSTOMER_ID,  
SUM(UNIT_PRICE * QUANTITY) AS REVENUE  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1  
ORDER BY 1;
```

```
43      -- 2. Find the total revenue for each customer.  
44  
45      SELECT CUSTOMER_ID,  
46      SUM(UNIT_PRICE * QUANTITY) AS REVENUE  
47      FROM EMP_DATABASE.PUBLIC.SALES  
48      GROUP BY 1  
49      ORDER BY 1;  
50
```

	CUSTOMER_ID	REVENUE
1	101	81.50
2	102	25.00
3	103	30.00
4	104	15.75
5	105	50.00
6	106	20.00
7	107	60.00
8	108	62.50
9	109	10.00

3. Get the products with more than 10 units sold in a single order.

```
SELECT PRODUCT_NAME, ORDER_ID,  
SUM(QUANTITY) AS SOLD_QTY  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1,2  
HAVING SOLD_QTY > 10;
```

So there is none such a record.

```
51      -- 3. Get the products with more than 10 units sold in a single order.  
52  
53      SELECT PRODUCT_NAME, ORDER_ID,  
54      SUM(QUANTITY) AS SOLD_QTY  
55      FROM EMP_DATABASE.PUBLIC.SALES  
56      GROUP BY 1,2  
57      HAVING SOLD_QTY > 10;  
58
```

PRODUCT_NAME	ORDER_ID	SOLD_QTY
--------------	----------	----------

Query produced no results

4. List the customers who have placed orders on at least three different dates.

```
SELECT CUSTOMER_ID, ORDER_ID,  
COUNT(DISTINCT ORDER_DATE) AS TOT_DIFF_DATE_ORDERS  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1,2  
HAVING TOT_DIFF_DATE_ORDERS >= 3;
```

So there is none such a record.

```
59 -- 4. List the customers who have placed orders on at least three different dates.  
60  
61 SELECT CUSTOMER_ID, ORDER_ID,  
62 COUNT(DISTINCT ORDER_DATE) AS TOT_DIFF_DATE_ORDERS  
63 FROM EMP_DATABASE.PUBLIC.SALES  
64 GROUP BY 1,2  
65 HAVING TOT_DIFF_DATE_ORDERS >= 3;
```

Results Chart

	CUSTOMER_ID	ORDER_ID	TOT_DIFF_DATE_ORDERS
--	-------------	----------	----------------------

Query produced no results

5. Calculate the average unit price of products.

```
SELECT PRODUCT_NAME,  
AVG(UNIT_PRICE) AS AVG_UNIT_PRICE  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1  
ORDER BY 1;
```

```
68 -- 5. Calculate the average unit price of products.  
69  
70 SELECT PRODUCT_NAME,  
71 AVG(UNIT_PRICE) AS AVG_UNIT_PRICE  
72 FROM EMP_DATABASE.PUBLIC.SALES  
73 GROUP BY 1  
74 ORDER BY 1;  
75
```

Results Chart

	PRODUCT_NAME	AVG_UNIT_PRICE
1	Widget A	10.00000000
2	Widget B	12.50000000
3	Widget C	15.75000000
4	Widget D	20.00000000

6. Find the products with an average unit price greater than \$12.00.

```
SELECT PRODUCT_NAME,  
AVG(UNIT_PRICE) AS AVG_UNIT_PRICE  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1  
HAVING AVG_UNIT_PRICE > 12.00  
ORDER BY 1;
```

```
76      -- 6. Find the products with an average unit price greater than $12.00.  
77  
78      SELECT PRODUCT_NAME,  
79      AVG(UNIT_PRICE) AS AVG_UNIT_PRICE  
80      FROM EMP_DATABASE.PUBLIC.SALES  
81      GROUP BY 1  
82      HAVING AVG_UNIT_PRICE > 12.00  
83      ORDER BY 1;
```

Results

Chart

	PRODUCT_NAME	AVG_UNIT_PRICE
1	Widget B	12.50000000
2	Widget C	15.75000000
3	Widget D	20.00000000

7. Retrieve the customers who have spent more than \$100.00 in total.

```
SELECT CUSTOMER_ID,  
SUM(QUANTITY * UNIT_PRICE) AS TOT_MONEY_SPENT  
FROM EMP_DATABASE.PUBLIC.SALES  
GROUP BY 1  
HAVING TOT_MONEY_SPENT > 100.00  
ORDER BY 1;
```

So there is none such a record.

```
85      -- 7. Retrieve the customers who have spent more than $100.00 in total.  
86  
87      SELECT CUSTOMER_ID,  
88      SUM(QUANTITY * UNIT_PRICE) AS TOT_MONEY_SPENT  
89      FROM EMP_DATABASE.PUBLIC.SALES  
90      GROUP BY 1  
91      HAVING TOT_MONEY_SPENT > 100.00  
92      ORDER BY 1;
```

Results

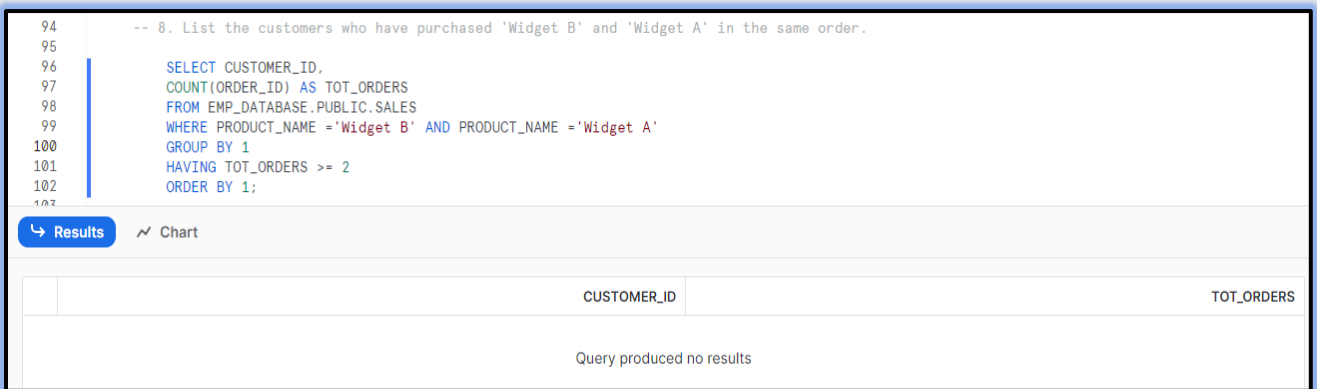
Chart

	CUSTOMER_ID	TOT_MONEY_SPENT
Query produced no results		

8. List the customers who have purchased 'Widget B' and 'Widget A' in the same order.

```
SELECT CUSTOMER_ID,  
COUNT(ORDER_ID) AS TOT_ORDERS  
FROM EMP_DATABASE.PUBLIC.SALES  
WHERE PRODUCT_NAME ='Widget B' AND PRODUCT_NAME ='Widget A'  
GROUP BY 1  
HAVING TOT_ORDERS >= 2  
ORDER BY 1;
```

So there is none such a record.



The screenshot shows a SQL query execution interface. The query is as follows:

```
-- 8. List the customers who have purchased 'Widget B' and 'Widget A' in the same order.  
  
SELECT CUSTOMER_ID,  
COUNT(ORDER_ID) AS TOT_ORDERS  
FROM EMP_DATABASE.PUBLIC.SALES  
WHERE PRODUCT_NAME ='Widget B' AND PRODUCT_NAME ='Widget A'  
GROUP BY 1  
HAVING TOT_ORDERS >= 2  
ORDER BY 1;
```

Below the query, there are two tabs: "Results" (selected) and "Chart". The "Results" tab shows a table with two columns: "CUSTOMER_ID" and "TOT_ORDERS". The table is empty, and a message below it states "Query produced no results".

CUSTOMER_ID	TOT_ORDERS
-------------	------------

Query produced no results

***** THANK YOU *****