SQL & Pandas – Mentoring Week 2
Shafa Aldiena Meraldin
shafaaldienameraldin@gmail.com

Find information about how many product have been ordered

1. (10 point) Show the product IDs and product names that have been ordered more than once! Sorted by product ID

Answer here

- **SQL Query Syntax**

```sql
SELECT product_id, product_name, count(order_id) as qty_ordered
FROM products
JOIN sales
USING ("product_id")
GROUP BY product_id, product_name
having count(order_id) > 1
order by product_id asc;
```

- **Screenshot of Query Results:**

	product_id	product_name	qty_ordered
1	1	Oxford Cloth	7
2	2	Oxford Cloth	7
3	3	Oxford Cloth	5
4	4	Oxford Cloth	7
5	5	Oxford Cloth	2
6	6	Oxford Cloth	5
7	7	Oxford Cloth	4
8	8	Oxford Cloth	5
9	9	Oxford Cloth	5
10	11	Oxford Cloth	3
11	12	Oxford Cloldh	2
12	13	Oxford Cloth	3
13	14	Oxford Cloth	2

1131	1,242	Tracksuit Bottoms	6
1132	1,244	Tracksuit Bottoms	3
1133	1,245	Tracksuit Bottoms	8
1134	1,246	Tracksuit Bottoms	4
1135	1,247	Tracksuit Bottoms	3
1136	1,248	Tracksuit Bottoms	3
1137	1,249	Tracksuit Bottoms	5
1138	1,250	Tracksuit Bottoms	8
1139	1,252	Tracksuit Bottoms	2
1140	1,253	Tracksuit Bottoms	6
1141	1,255	Tracksuit Bottoms	5
1142	1,256	Tracksuit Bottoms	3
1143	1,257	Tracksuit Bottoms	3
1144	1,258	Tracksuit Bottoms	4
1145	1,259	Tracksuit Bottoms	4

- **Description of Query Results:**
  In this query, we retrieved a total of 1145 data from the products that have been ordered more than once.

2. (10 point) From question number 1, How many products have been ordered more than once?

Answer here

- **SQL Query Syntax**

```sql
WITH ordered_products AS (
    SELECT product_id, product_name, COUNT(order_id) AS qty_ordered
    FROM products
    JOIN sales USING (product_id)
    GROUP BY product_id, product_name
    HAVING COUNT(order_id) > 1
)
SELECT COUNT(*) AS total_products
FROM ordered_products;
```

- **Screenshot of Query Results:**

	total_products
1	1,145

- **Description of Query Results:**
  In this query, we can conclude that there are 1145 products have been ordered more than once.

3. (10 point) From question number 2, How many products have only been ordered once?
   Answer here
   - **SQL Query Syntax**

```sql
WITH ordered_products AS (
    SELECT product_id, product_name, COUNT(order_id) AS qty_ordered
    FROM products
    JOIN sales USING (product_id)
    GROUP BY product_id, product_name
    HAVING COUNT(order_id) = 1
)
SELECT COUNT(*) AS total_products
FROM ordered_products;
```

   - **Screenshot of Query Results:**

	123 total_products
1	88

   - **Description of Query Results:**
     In this query, we can conclude that there are 88 products have only been ordered once.

Next, Dig deeper information into customer behavior

4. (10 point) list of customers who have placed orders more than twice in a single month. Manager need customer name and their address to give the customer special discount
   Answer here
   - **SQL Query Syntax**

```sql
SELECT customer_id, customer_name, home_address,
       EXTRACT(YEAR FROM CAST(order_date AS date)) AS order_year,
       EXTRACT(MONTH FROM CAST(order_date AS date)) AS order_month,
       count(order_id) as total_orders
FROM customers
JOIN orders
USING (customer_id)
GROUP BY     customer_id, customer_name, home_address,
             EXTRACT(YEAR FROM CAST(order_date AS date)),
             EXTRACT(MONTH FROM CAST(order_date AS date))
HAVING COUNT(order_id) > 2;
```

   - **Screenshot of Query Results:**

	123 customer_id	A-Z customer_name	A-Z home_address	123 order_year	123 order_month	123 total_orders
1	526	Ellyn Collacombe	786 Paige CircleApt. 922	2,021	8	3
2	732	Gorden Seago	905 Stephanie BoulevardApt. 217	2,021	4	3

   - **Description of Query Results:**
     In this query, we identified 2 customers who qualified for a special discount based on their order frequency. Based on the output, Ellyn Collacombe (Cust ID: 526) from 786 Paige Circle Apt. 922 placed 3 orders in August 2021, while Gorden Seago (Cust ID: 732) from 905 Stephanie Boulevard Apt. 217 placed 3 orders in April 2021.

5. (10 point) Find the first and last order date of each customer. Show the first 10 data, sorted by customer ID
   Answer here

- **SQL Query Syntax**

```
SELECT        customer_id,
       MIN(CAST(order_date AS date)) AS first_order_date,
       MAX(CAST(order_date AS date)) AS last_order_date
from orders
group by customer_id
order by customer_id asc
limit 10;
```

- **Screenshot of Query Results:**

	customer_id	first_order_date	last_order_date
1	1	2021-02-18	2023-01-15
2	2	2023-01-16	2023-01-16
3	3	2023-01-18	2023-01-18
4	4	2023-01-19	2023-01-19
5	5	2023-01-20	2023-01-20
6	7	2021-05-21	2021-05-21
7	10	2021-03-09	2021-03-09
8	11	2021-05-28	2021-05-28
9	12	2021-06-19	2021-06-19
10	13	2021-09-28	2021-09-28

- **Description of Query Results:**
- The query retrieves the first and last order dates for each customer limited to the first 10 records. Based on the output, Customer 1 had their first order on February 18, 2021, and their last order on January 15, 2023, indicating multiple purchases over time. In contrast, customers 2 to 13 (excluding customer 6, 8, and 9, who are not in the top 10 because they have not placed any orders) had the same first and last order date, suggesting they made a single purchase.

6. (10 point) Retrieve the top 5 customers who have spent the most amount of money on products within the “Trousers” category, including the customer's name, the quantity and total amount spent in this category. Additionally, find the total number of products sold in this category and calculate the average total price spent in this category, compare with the top 5 customers who have spent the most amount of money on products within the “Trousers” category . Finally, sort the results by the total amount spent in descending order.

Answer here

- **SQL Query Syntax**

```
with most_spender as (
   select     customer_id, customer_name,
              sum(sales.quantity * sales.price_per_unit) as total_spent,
              sum(sales.quantity) as qty_order
   from customers
   join orders
   using (customer_id)
   join sales
   using (order_id)
   join products
   using (product_id)
   where product_type = 'Trousers'
   group by customer_id, customer_name
   order by total_spent desc
   limit 5
),
data_trousers as (
```

```
  select     sum(sales.quantity) as qty_trousers,
             avg(sales.quantity * sales.price_per_unit) as avg_price_trousers
  from sales
  join products
  using (product_id)
  where product_type = 'Trousers'
)
select       ms.customer_id,
       ms.customer_name,
       ms.total_spent,
       ms.qty_order,
       dt.qty_trousers,
       dt.avg_price_trousers
from most_spender as ms
cross join data_trousers as dt
order by total_spent desc;
```

- **Screenshot of Query Results:**

	customer_id	customer_name	total_spent	qty_order	qty_trousers	avg_price_trousers
1	571	Kristofor Roos	2,827	28	3,360	202.7177658942
2	348	Thorny Nornable	2,802	28	3,360	202.7177658942
3	163	Harrietta Burchatt	2,426	24	3,360	202.7177658942
4	282	Wren Helgass	2,272	22	3,360	202.7177658942
5	465	Mallory Castellani	2,126	21	3,360	202.7177658942

- **Description of Query Results:**

  In this query, we identified the top 5 customers who have spent the most money on Trousers, while also providing insights into the total quantity sold and the average total price spent in this category.

  From the results:

  - Kristofor Roos (Cust ID: 571) spent the most at $2,827, purchasing 28 trousers.
  - Thorny Nornable (Cust ID: 348) followed closely, spending $2,802 on 28 trousers.
  - Harrietta Burchatt (Cust ID: 163) spent $2,426 for 24 trousers.
  - Wren Helgass (Cust ID: 282) spent $2,272 for 22 trousers.
  - Mallory Castellani Cust (ID: 465) spent $2,126 for 21 trousers.

  In comparison, across all customers, the total number of trousers sold was 3,360, and the average total price of trousers was $202.72.

  The results indicate that the top spenders purchased significantly more than the average, reinforcing their status as high-value customers for this product category.

7. (10 point) Find the top-selling (Top 1) product for each month. You want to know the product with the highest **total quantity sold** in each month. If there are products that have the same total quantity sold, choose the smallest product ID. Return the product name, the corresponding month, and the total quantity sold for each month's top-selling product. Sorted by month

Answer here

- **SQL Query Syntax**

```
with monthly_sales as (
      select  EXTRACT(MONTH FROM CAST(order_date AS date)) AS sale_month,
                    EXTRACT(YEAR FROM CAST(order_date AS date)) AS sale_year,
                    product_id, product_name,
                    SUM(sales.quantity) as total_quantity
      from products
      join sales
      using (product_id)
```

```
      join orders
      using (order_id)
      where order_date is not null
      group by sale_month, sale_year, product_id, product_name
),
product_ranking as (
      select *,
            RANK() over (partition by sale_month, sale_year
                         order by total_quantity desc, product_id ASC) as pr
      from monthly_sales
)
select sale_month, sale_year, product_id, product_name, total_quantity
from product_ranking
where pr = 1
order by sale_year, sale_month;
```

- **Screenshot of Query Results:**

	123 sale_month	123 sale_year	123 product_id	A-Z product_name	123 total_quantity
1	1	2,021	1,084	Joggers	10
2	2	2,021	920	Drawstring	9
3	3	2,021	1	Oxford Cloth	10
4	4	2,021	850	Chinos	8
5	5	2,021	650	Coach	11
6	6	2,021	1,139	Cargo Pants	10
7	7	2,021	383	Henley	7
8	8	2,021	125	Denim	7
9	9	2,021	28	Oxford Cloth	9
10	10	2,021	1,177	High-Waisted	9

- **Description of Query Results:**
  From this query, we can identified that the dataset captured orders during the period between January 2021 till October 2021 and here are the results that we can concluded :
  - In January 2021, Joggers (Prod ID: 1084) was the top seller with 10 units sold.
  - In February 2021, Drawstring (Prod ID: 920) was the top seller with 9 units sold.
  - In March 2021, Oxford Cloth (Prod ID: 1) was the top seller with 10 units sold.
  - In May 2021, Coach (Prod ID: 650) had the highest sales with 11 units sold, making it the highest monthly quantity sold in the dataset.
  - In June 2021, Cargo Pants (Prod ID: 1139) was the top seller with 10 units sold.
  - In July 2021, Henley (Prod ID: 383) was the top seller with 7 units sold.
  - In August 2021, Denim was the top seller with 7 units sold.
  - In September 2021, Oxford Cloth was the top seller with 9 units sold.
  - In October 2021, High-Waisted was the top seller with 9 units sold.

8. (10 point) Create a view to store a query for calculating monthly total payment.
   Answer here

- **SQL Query Syntax**

```
CREATE VIEW monthlytotalpayment AS
SELECT
    EXTRACT(YEAR FROM CAST(order_date AS DATE)) AS sale_year,
    EXTRACT(MONTH FROM CAST(order_date AS DATE)) AS sale_month,
    SUM(payment) AS total_transaction_amount
FROM orders
GROUP BY sale_year, sale_month
ORDER BY sale_month, sale_year asc;
```

```sql
SELECT * FROM monthlytotalpayment;
```

- **Screenshot of Query Results:**

Name	Value
Updated Rows	0
Execute time	0.035s
Start time	Fri Feb 28 16:07:56 WIB 2025
Finish time	Fri Feb 28 16:07:56 WIB 2025
Query	CREATE VIEW monthlytotalpayment AS SELECT EXTRACT(YEAR FROM CAST(order_date AS DATE)) AS sale_year, EXTRACT(MONTH FROM CAST(order_date AS DATE)) AS sale_month, SUM(payment) AS total_transaction_amount FROM orders GROUP BY sale_year, sale_month ORDER BY sale_month, sale_year asc

	123 sale_year	123 sale_month	123 total_transaction_amount
1	2,021	1	3,520,014
2	2,023	1	540
3	2,021	2	3,190,783
4	2,021	3	4,129,045
5	2,021	4	3,233,260
6	2,021	5	3,028,007
7	2,021	6	3,351,443
8	2,021	7	3,588,328
9	2,021	8	3,822,263
10	2,021	9	3,173,182
11	2,021	10	2,936,611

- **Description of Query Results:**

From this query and view, we can identified results below :

- January 2021 recorded the highest total payment at 3,520,014, whereas January 2023 had a significantly lower total of 540, indicating fewer transactions in that period.
- March 2021 had the highest total payment of 4,129,045, suggesting peak sales activity.
- Throughout 2021, the monthly total payments varied, with notable highs in August (3,822,263) and July (3,588,328).
- The total payment values show fluctuations, potentially indicating seasonal trends or changes in customer purchasing behavior.

9. (10 point) As a warehouse manager responsible for stock management in your company's warehouse, you oversee a warehouse with a total area of 600,000 sq ft. There are two types of items: prime items and non-prime items. These items come in various sizes, with priority given to prime items. Your task is to determine the maximum number of prime and non-prime items that can be stored in the warehouse

- Prime and non-prime items are stored in their respective containers. For example, In the database, there are 15 non-prime items and 35 prime items. Each prime container must contain 35 prime items, and each non-prime container must contain 15 non-prime items
- Non-prime items must always be available in stock to meet customer demand, so the non-prime item count should never be zero.

Answer Here

- **SQL Query Syntax :**

```sql
-- cek data prime and non prime
select *
from item

-- total kapasitas penyimpanan adalah 600.000
select sum(item_size_sqft)
from item

-- list item prime
create temp table item_prime as
select       sum(item_size_sqft) as prime_container_size,
       count(item_size_sqft) as qty_prime
from item
where is_prime is true;
```

```sql
-- Mengecek hasil dalam temporary table
SELECT * FROM item_prime;

-- list item non prime
create temp table item_nonprime as
select       sum(item_size_sqft) as nonprime_container_size,
       count(item_size_sqft) as qty_nonprime
from item
where is_prime is false;

-- Mengecek hasil dalam temporary table
SELECT * FROM item_nonprime;

-- jumlah kontainer prime
create temp table cont_prime as
select  floor(600000.0 / prime_container_size) as qty_container_prime,
       qty_prime
from item_prime;

-- jml item prime
create temp table item_summary_prime as
select qty_container_prime * qty_prime as item_count_prime
from cont_prime;


-- jumlah kontainer non prime
create temp table cont_nonprime as
with sisa_luas_warehouse as (
select (600000.0 - qty_container_prime * prime_container_size) as
sisa_luas_warehouse
from item_prime
cross join cont_prime
)
select floor(sisa_luas_warehouse / nonprime_container_size) as
qty_container_nonprime,
       qty_nonprime
from sisa_luas_warehouse
cross join item_nonprime;


-- juml item nonpime
create temp table item_summary_nonprime as
select qty_container_nonprime * qty_nonprime as item_count_nonprime
from cont_nonprime;

-- Gabungkan hasil prime dan non-prime ke dalam satu query
select 'prime items' as is_prime, item_count_prime as item_count
from item_summary_prime

union all

select 'non-prime items' as is_prime, qty_container_nonprime * qty_nonprime as
item_count
from cont_nonprime;
```

- **Screenshot of Query Results:**

	is_prime	item_count
1	prime items	735
2	non-prime items	30

- **Description of Query Results:**
  From this query, the output shows that 735 prime items are allocated more space due to their priority, while 30 non-prime items are stored to ensure continuous availability. The query determines the maximum number of prime and non-prime items that can be stored within a 600,000 sq ft warehouse while prioritizing prime items. It calculates the total storage occupied by prime and non-prime items, then determines how many containers of each type can fit within the available space. Prime items are given priority, so their containers are allocated first. The remaining warehouse space is then used for non-prime items, ensuring that at least one non-prime container is always available. By calculating storage distribution efficiently, this query helps warehouse managers optimize space utilization while maintaining minimum stock levels for non-prime items.

10. (10 point) The warehouse manager is planning to find a new warehouse to store their products. The warehouse is expected to accommodate 20 containers for each prime and non-prime item. What is the minimum required size for the warehouse?

- **SQL Query Syntax :**

```
-- new warehouse for 20 container prime and 20 container non prime
-- total new warehouse area
select 20 * prime_container_size + 20 * nonprime_container_size as
total_wh_area
from item_prime
cross join item_nonprime;
```

- **Screenshot of Query Results:**

	total_wh_area
1	670,800

- **Description of Query Results:**
  From this query we can determined the total warehouse area required. The final output, 670,800 sq ft, represents the necessary warehouse space, ensuring adequate capacity for storing 20 prime containers and 20 non-prime containers.