

Data Engineering 101 SQL Interview Questions: Day 1



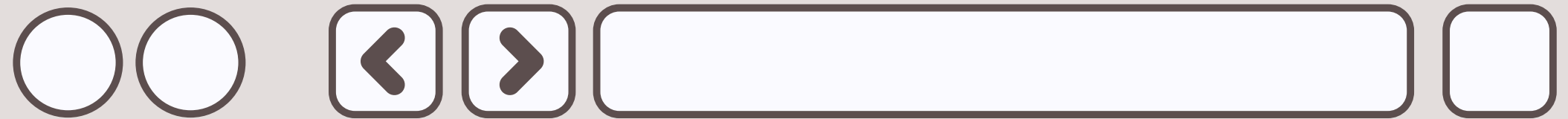
Shwetank Singh
GritSetGrow - GSGLearn.com

WRITE A QUERY TO FIND THE TOP 3 CUSTOMERS WHO HAVE MADE THE MOST PURCHASES IN THE LAST MONTH.

```
WITH customer_purchases AS (  
  SELECT customer_id, COUNT(*) as purchase_count,  
         ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) as row_num  
  FROM orders  
  WHERE order_date >= DATE_TRUNC('month', CURRENT_DATE) - INTERVAL '1  
month'  
  GROUP BY customer_id  
)  
SELECT customer_id, purchase_count  
FROM customer_purchases  
WHERE row_num <= 3;
```



CREATE A QUERY TO CALCULATE A RUNNING TOTAL OF SALES FOR EACH PRODUCT CATEGORY, ORDERED BY DATE.



```
SELECT
  o.order_date, p.category, SUM(o.total_amount) as daily_sales,
  SUM(SUM(o.total_amount)) OVER (
    PARTITION BY p.category ORDER BY o.order_date
    ROWS UNBOUNDED PRECEDING
  ) as running_total
FROM
  orders o
JOIN
  products p ON o.product_id = p.id
GROUP BY
  o.order_date, p.category
ORDER BY
  p.category, o.order_date;
```



WRITE A QUERY TO FIND EMPLOYEES WHO EARN MORE THAN THEIR DEPARTMENT'S AVERAGE SALARY.

```
WITH dept_avg AS (  
    SELECT department_id, AVG(salary) as avg_salary  
    FROM employees  
    GROUP BY department_id  
)  
SELECT e.employee_id, e.name, e.salary, e.department_id  
FROM employees e  
JOIN dept_avg d ON e.department_id = d.department_id  
WHERE e.salary > d.avg_salary;
```



CREATE A QUERY TO IDENTIFY CUSTOMERS WHO HAVE MADE PURCHASES IN CONSECUTIVE MONTHS.

```
WITH monthly_purchases AS (  
  SELECT  
    customer_id,  
    DATE_TRUNC('month', order_date) as order_month,  
    LEAD(DATE_TRUNC('month', order_date), 1) OVER (  
      PARTITION BY customer_id  
      ORDER BY DATE_TRUNC('month', order_date)  
    ) as next_month  
  FROM orders  
)  
SELECT DISTINCT customer_id  
FROM monthly_purchases  
WHERE DATEDIFF(month, order_month, next_month) = 1;
```



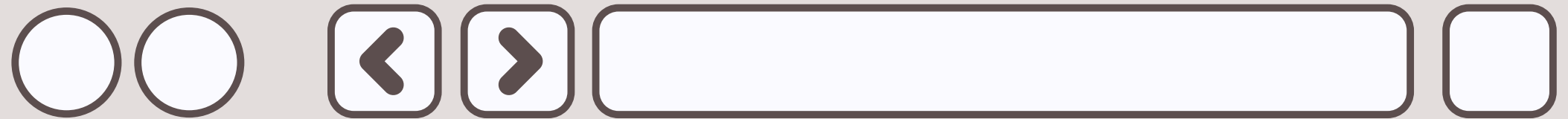
WRITE A QUERY TO PIVOT SALES DATA FROM ROWS TO COLUMNS, SHOWING QUARTERLY SALES FOR EACH PRODUCT.



```
SELECT
  product_id,
  SUM(CASE WHEN EXTRACT(QUARTER FROM order_date) = 1
    THEN total_amount ELSE 0 END) as Q1,
  SUM(CASE WHEN EXTRACT(QUARTER FROM order_date) = 2
    THEN total_amount ELSE 0 END) as Q2,
  SUM(CASE WHEN EXTRACT(QUARTER FROM order_date) = 3
    THEN total_amount ELSE 0 END) as Q3,
  SUM(CASE WHEN EXTRACT(QUARTER FROM order_date) = 4
    THEN total_amount ELSE 0 END) as Q4
FROM
  orders
WHERE
  EXTRACT(YEAR FROM order_date) = 2023
GROUP BY
  product_id;
```



CREATE A QUERY TO FIND THE MEDIAN SALARY FOR EACH DEPARTMENT.



```
WITH ranked_salaries AS (  
  SELECT department_id, salary,  
         ROW_NUMBER()  
           OVER (PARTITION BY department_id ORDER BY salary) as row_num,  
         COUNT(*) OVER (PARTITION BY department_id) as dept_count  
  FROM employees  
)  
SELECT department_id, AVG(salary) as median_salary  
FROM  
  ranked_salaries  
WHERE  
  row_num IN (FLOOR((dept_count+1)/2), CEIL((dept_count+1)/2))  
GROUP BY  
  department_id;
```



WRITE A QUERY TO FIND THE TOP PRODUCT IN EACH CATEGORY BASED ON TOTAL SALES AMOUNT.

```
WITH ranked_products AS (  
    SELECT p.category, p.product_id, SUM(o.total_amount) as total_sales,  
           RANK() OVER (PARTITION BY p.category ORDER BY SUM(o.total_amount)  
DESC) as rank  
    FROM  
        products p  
    JOIN  
        orders o ON p.product_id = o.product_id  
    GROUP BY  
        p.category, p.product_id  
)  
SELECT category, product_id, total_sales  
FROM ranked_products  
WHERE rank = 1;
```



CREATE A QUERY TO CALCULATE THE YEAR-OVER-YEAR GROWTH RATE FOR EACH PRODUCT.

```
WITH yearly_sales AS (  
    SELECT EXTRACT(YEAR FROM order_date) as year, product_id,  
           SUM(total_amount) as yearly_total  
    FROM orders  
    GROUP BY EXTRACT(YEAR FROM order_date), product_id  
)  
SELECT current.year, current.product_id, current.yearly_total,  
       previous.yearly_total as prev_year_total,  
       (current.yearly_total - previous.yearly_total) / previous.yearly_total * 100 as growth_rate  
FROM yearly_sales current  
LEFT JOIN  
    yearly_sales previous ON current.product_id = previous.product_id  
                        AND current.year = previous.year + 1  
WHERE  
    previous.yearly_total IS NOT NULL;
```

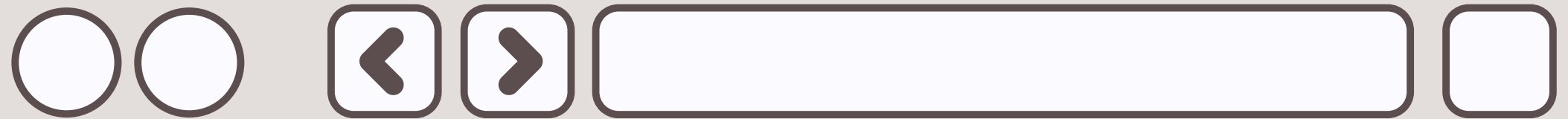


WRITE A QUERY TO IDENTIFY CUSTOMERS WHO HAVE NEVER MADE A PURCHASE.

```
SELECT c.customer_id, c.name  
FROM customers c  
LEFT JOIN orders o ON c.customer_id = o.customer_id  
WHERE o.order_id IS NULL;
```



CREATE A QUERY TO CALCULATE THE RUNNING TOTAL OF INVENTORY FOR EACH PRODUCT, CONSIDERING BOTH ADDITIONS AND SUBTRACTIONS.



```
WITH inventory_changes AS (  
    SELECT product_id, change_date, quantity,  
           SUM(quantity) OVER (PARTITION BY product_id ORDER BY change_date) as  
running_total  
    FROM (  
        SELECT product_id, date as change_date, received_quantity as quantity  
        FROM inventory_receipts  
        UNION ALL  
        SELECT product_id, date as change_date, -shipped_quantity as quantity  
        FROM inventory_shipments  
    ) all_changes  
)  
SELECT product_id, change_date, quantity, running_total  
FROM inventory_changes  
ORDER BY product_id, change_date;
```



THANK you

