**Day 18 - SQL Questions**

**Question:** Write an SQL query that selects the product id, year, quantity, and price for the first year of every product sold. Return the resulting table in any order.

**Answer:**

First I tried this. The output was correct, but the answer was rejected.

WITH first\_year\_sales AS(

SELECT

product\_id, year, quantity, price,

RANK() OVER (

PARTITION BY year

) year\_rank

FROM

Sales

GROUP BY product\_id

)

SELECT product\_id, year as first\_year, quantity, price

FROM first\_year\_sales

WHERE year\_rank = 1

This was someone else’s answer - it was accepted:

SELECT product\_id , year as first\_year , quantity , price

FROM Sales

WHERE (product\_id , year)

IN

(

# Write your MySQL query statement below

SELECT product\_id , MIN(year) as year from Sales

GROUP BY product\_id

)

This was NOT accepted although again output was identical to other two:

SELECT product\_id, MIN(year) as first\_year, quantity, price

FROM Sales

GROUP BY product\_id

**Question:**

Where Table: Products

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| product\_id | int |

| store | enum |

| price | int |

+-------------+---------+

(product\_id, store) is the primary key for this table.

store is an ENUM of type ('store1', 'store2', 'store3') where each represents the store this product is available at.

price is the price of the product at this store.

Write an SQL query to find the price of each product in each store.

Return the result table in any order.

**Answer:**

I needed someone else’s example to help me get this right. I was on the right track with a subquery for price by store, but did not come up with using a CASE statement for this on my own.

SELECT

product\_id,

MAX(CASE WHEN store = 'store1' THEN price ELSE NULL END) AS store1,

MAX(CASE WHEN store = 'store2' THEN price ELSE NULL END) AS store2,

MAX(CASE WHEN store = 'store3' THEN price ELSE NULL END) AS store3

FROM Products

GROUP BY product\_id

**Question:**

Beginning with Table: Products

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| product\_id | int |

| store1 | int |

| store2 | int |

| store3 | int |

+-------------+---------+

product\_id is the primary key for this table.

Each row in this table indicates the product's price in 3 different stores: store1, store2, and store3.

If the product is not available in a store, the price will be null in that store's column.

Write an SQL query to rearrange the Products table so that each row has (product\_id, store, price). If a product is not available in a store, do not include a row with that product\_id and store combination in the result table.

Return the result table in any order.

**Answer:**

I don’t well-remember using UNION, but discovered by looking at other people’s answers that this problem called for it. After looking at the answer, it seemed simple! Many people commented that unlike in python, you cannot ‘pivot’ from cols to rows in SQL. Good information.

SELECT product\_id, 'store1' AS store, store1 AS price

FROM Products

WHERE store1 IS NOT NULL

UNION

SELECT product\_id, 'store2' AS store, store2 AS price

FROM Products

WHERE store2 IS NOT NULL

UNION

SELECT product\_id, 'store3' AS store, store3 AS price

FROM Products

WHERE store3 IS NOT NULL