SQL-04 | Aggregation

Lecture Queries

Question: Given a houses table and a Renters table, list all the possible pairs of renters along with their preferred districts to see which renters could potentially rent a house together.

`Houses`

id	district	address	bedrooms	rent
1	South	Rose Street, 5	4	3000.00
2	North	Main Street, 12	3	2250.00
3	South	Rose Street, 5	4	3000.00
4	West	Nice Street, 3	2	1750.00

`Renters`

id	name	preferred_district	min_bedrooms	min_rent	max_rent
1	Helen Boss	South	3	2500.00	3200.00
2	Michael Lane	West	2	1500.00	2500.00
3	Susan Sanders	West	4	2500.00	4000.00
4	Tom White	North	3	2200.00	2500.00

Question: Given a houses table and a Renters table, list all the possible pairs of renters along with their preferred districts to see which renters could potentially rent a house together.

```
SELECT r1.name, r1.preferred_district, r2.name,
r2.preferred_district
FROM renters r1
JOIN renters r2
ON r1.preferred_district = r2.preferred_district AND r1.id != r2.id;
```

Range of values

Question: Recommend houses to renters that are

(1) in their preferred district,

(2) within their price range,

(3) with their required number of bedrooms,

```
SELECT r.id, r.name, h.id, h.address, h.rent, h.bedrooms
FROM renters r
JOIN houses h
ON h.district = r.preferred_district
   AND h.rent BETWEEN r.min_rent AND r.max_rent
   AND h.bedrooms >= r.min_bedrooms;
```

Question: Suppose we want to find all employees with a salary higher than their average **departmental salary.**

Correlated Subqueries

Correlated subquery: It gets its name because the two queries are related; the inner query uses information obtained from the outer query

We need a temporary table to calculate the average for every department. And for every department, we need to ensure that the department of employees should be same.

```
SELECT
  lastname,
  firstname,
  salary
FROM employee e1
WHERE e1.salary > (SELECT avg(salary)
                   FROM employee e2
                   WHERE e2.dept_id = e1.dept_id)
```

Question: Get a list of customer IDs of customers who made purchases on each market date.

```
SELECT
    market_date,
    customer_id
 FROM farmers_market.customer_purchases
 GROUP BY market_date, customer_id
 ORDER BY market_date, customer_id
```

Question: Count the number of purchases each customer made per market date.

Question: Count the number of purchases each customer made per market date.

```
SELECT
    market_date,
    customer_id,
    COUNT(*) AS num_purchases
FROM farmers_market.customer_purchases
GROUP BY market_date, customer_id
ORDER BY market_date, customer_id
LIMIT 10
```

Question: Calculate the total quantity that each customer bought per market date.

Question: Calculate the total quantity that each customer bought per market date.

```
SELECT
    market_date,
    customer_id,
    SUM(quantity) AS total_qty_purchased
FROM farmers_market.customer_purchases
GROUP BY market_date, customer_id
ORDER BY market_date, customer_id
```

Question: how many different kinds of products were purchased by each customer on each market date:

Question: how many different kinds of products were purchased by each customer on each market date:

```
SELECT
    market_date,
    customer_id,
    COUNT(DISTINCT product_id) AS different_products_purchased
FROM farmers_market.customer_purchases
GROUP BY market_date, customer_id
ORDER BY market_date, customer_id
```

Question: Calculate the total price paid by customer_id 3 per market_date.

Question: Calculate the total price paid by customer_id 3 per market_date.

```
SELECT
        customer_id,
        market_date,
        SUM(quantity * cost_to_customer_per_qty) AS total_spent
    FROM farmers_market.customer_purchases
    WHERE
        customer_id = 3
    GROUP BY market_date
    ORDER BY market_date
```

Question: What if we wanted to find out how much a customer had spent at each vendor, regardless of date?

Question: Let’s add some customer details and vendor details to these results. Customer details are in the customer table and vendor details are in the vendor table.

Question: Let's add some customer details and vendor details to these results.
Customer details are in the customer table and vendor details are in the vendor table.

```
SELECT
    c.customer_first_name,
    c.customer_last_name,
    cp.customer_id,
    v.vendor_id,
    v.vendor_name,
    SUM(quantity * cost_to_customer_per_qty) AS total_price
FROM customer AS c
LEFT JOIN customer_purchases AS cp
    ON c.customer_id = cp.customer_id
LEFT JOIN vendor AS v
    ON cp.vendor_id = v.vendor_id
GROUP BY
    cp.customer_id,
    v.vendor_id,
```

Question: We want to get the most and least expensive items per product category, considering the fact that each vendor sets their own prices and can adjust prices per customer.

Question: We want to get the most and least expensive items per product category, considering the fact that each vendor sets their own prices and can adjust prices per customer.

```
SELECT
    pc.product_category_name,
    p.product_category_id,
    MIN(vi.original_price) AS minimum_price,
    MAX(vi.original_price) AS maximum_price
FROM farmers_market.vendor_inventory AS vi
    INNER JOIN farmers_market.product AS p
        ON vi.product_id = p.product_id
    INNER JOIN farmers_market.product_category AS pc
        ON p.product_category_id = pc.product_category_id
GROUP BY pc.product_category_name, p.product_category_id
```

Question: filter out vendors who brought at least 10 items to the farmer’s market over the time period - 2019-05-02 and 2019-05-16

Question: filter out vendors who brought at least 10 items to the farmer's market over the time period - 2019-05-02 and 2019-05-16

```
SELECT
    vendor_id,
    COUNT(DISTINCT product_id) AS different_products_offered,
    SUM(quantity * original_price) AS value_of_inventory,
    SUM(quantity) AS inventory_item_count,
    SUM(quantity * original_price) / SUM(quantity) AS average_item_price
FROM farmers_market.vendor_inventory
WHERE market_date BETWEEN '2019-05-02' AND '2019-05-16'
GROUP BY vendor_id
HAVING inventory_item_count >= 10
ORDER BY vendor_id
```

Question: Find the average amount spent by customers on each day. We want to consider only those days where the number of purchases were more than 3 and the transaction amount was greater than 5.

```
SELECT
  market_date,
  ROUND(AVG(quantity * cost_to_customer_per_qty), 2) AS
avg_spent
FROM customer_purchases
WHERE quantity * cost_to_customer_per_qty > 5
GROUP BY market_date
HAVING COUNT(*) > 3
ORDER BY market_date;
```

Reference Material

- COUNT(*) vs COUNT(1) vs COUNT(col_name)