이 쿼리들은 PostgreSQL에서 집계 함수(aggregation function)를 사용하여 데이터 집계를 수행하는 다양한 예제들입니다. 아래에 각 쿼리에 대한 설명을 제공하겠습니다.

1. 회원 테이블에서 전체 행 갯수 구하기

```
SELECT COUNT(*)
      ,COUNT(cust_id)
      ,COUNT(city)
      ,COUNT(region)
FROM customer;
```

- **COUNT(*)**: 전체 행의 수를 셉니다.
- **COUNT(column_name)**: 지정한 열의 값이 NULL이 아닌 행의 수를 셉니다.
- 여기서는 `cust_id`, `city`, `region`에 대해 각각 NULL이 아닌 값들의 개수를 셉니다.

2. 회원 테이블에서 마일리지 총합, 평균, 최소, 최대 구하기

```
SELECT SUM(mileage)
      ,AVG(mileage)
      ,MIN(mileage)
      ,MAX(mileage)
FROM customer;
```

- **SUM(mileage)**: 마일리지의 총합을 계산합니다.
- **AVG(mileage)**: 마일리지의 평균을 계산합니다.
- **MIN(mileage)**: 마일리지의 최소값을 반환합니다.
- **MAX(mileage)**: 마일리지의 최대값을 반환합니다.

3. 서울에 거주하는 회원의 마일리지 집계

```
SELECT SUM(mileage)
      ,AVG(mileage)
      ,MIN(mileage)
      ,MAX(mileage)
FROM customer
WHERE city LIKE '서울%';
```

- 서울로 시작하는 도시(`city LIKE '서울%'`)에 거주하는 회원들의 마일리지를 집계합니다.

4. 도시별 회원 수와 평균 마일리지 구하기

```
SELECT city
      ,COUNT(*) AS customer_count
      ,AVG(mileage) AS average_mileage
FROM customer
GROUP BY city;
```

- **GROUP BY city**: 도시별로 그룹화하여 각 도시의 회원 수와 평균 마일리지를 계산합니다.

5. 직급별 도시별 회원 수와 평균 마일리지 구하기

```
SELECT contact_position
      ,city
      ,COUNT(*) AS customer_count
      ,AVG(mileage) AS average_mileage
FROM customer
GROUP BY contact_position, city
ORDER BY contact_position, city;
```

- **GROUP BY contact_position, city**: 직급과 도시별로 그룹화하여 회원 수와 평균 마일리지를 계산합니다.
- **ORDER BY contact_position, city**: 결과를 직급과 도시 기준으로 정렬합니다.

6. 10명 이상의 고객이 있는 도시의 회원 수와 평균 마일리지

```
SELECT city
      ,COUNT(*) AS customer_count
      ,AVG(mileage) AS average_mileage
FROM customer
GROUP BY city
HAVING COUNT(*) >= 10;
```

- **HAVING COUNT(*) >= 10**: 고객 수가 10명 이상인 도시만 선택하여 회원 수와 평균 마일리지를 계산합니다.

7. 특정 조건을 만족하는 도시별 총 마일리지 구하기

```
SELECT city
      ,SUM(mileage)
FROM customer
WHERE company_name LIKE '%T%'
```

```
GROUP BY city
HAVING SUM(mileage) >= 100;
```

- 회사 이름에 'T'가 포함되고, 도시별 마일리지 합계가 100 이상인 경우에만 도시별 총 마일리지를 계산합니다.

8. 지역이 NULL이 아닌 도시별 고객 수 및 평균 마일리지

```
SELECT city
      ,COUNT(*) AS customer_count
      ,AVG(mileage) AS average_mileage
FROM customer
WHERE region IS NOT NULL
GROUP BY city;
```

- 지역이 NULL이 아닌 고객들만 대상으로 도시별 고객 수와 평균 마일리지를 계산합니다.

9. 직급이 '대표'인 고객의 도시별 고객 수

```
SELECT contact_position
      ,city
      ,COUNT(*) AS customer_count
FROM customer
WHERE contact_position LIKE '%대표%'
GROUP BY contact_position, city;
```

- 직급이 '대표'인 고객들을 대상으로 도시별 고객 수를 계산합니다.

10. 직급이 '대표'인 고객의 지역별 고객 수

```
SELECT region
      ,COUNT(*) AS customer_count
FROM customer
WHERE contact_position LIKE '%대표%'
GROUP BY region;
```

- 직급이 '대표'인 고객들을 대상으로 지역별 고객 수를 계산합니다.

11. 사원 테이블에서 이름을 콤마로 연결하여 출력하기

```sql
SELECT STRING_AGG(name, ', ') AS names
FROM employee;
```

- **STRING_AGG(name, ', ')**: 모든 사원의 이름을 콤마(`, `)로 연결하여 하나의 문자열로 반환합니다.

12. 회원 테이블에서 지역을 콤마로 연결하여 출력하기

```sql
SELECT STRING_AGG(DISTINCT region, ', ') AS regions
FROM customer;
```

- **STRING_AGG(DISTINCT region, ', ')**: 중복을 제거한 모든 지역 이름을 콤마(`, `)로 연결하여 하나의 문자열로 반환합니다.

13. 도시별 회사 이름을 콤마로 연결하여 출력하기

```sql
SELECT city
      ,STRING_AGG(company_name, ', ') AS customer_companies
FROM customer
GROUP BY city;
```

- 도시별로 회사 이름들을 콤마(`, `)로 연결하여 출력합니다.

14. 도시의 총 갯수와 중복을 제거한 도시의 개수 출력하기

```sql
SELECT COUNT(city) AS total_city_count
      ,COUNT(DISTINCT city) AS distinct_city_count
FROM customer;
```

- **COUNT(city)**: 전체 도시 수를 계산합니다.
- **COUNT(DISTINCT city)**: 중복을 제거한 도시 수를 계산합니다.

15. 주문 테이블에서 연도별 주문 수 출력하기

```sql
SELECT EXTRACT(YEAR FROM order_date) AS order_year
      ,COUNT(*) AS order_count
FROM orders
GROUP BY EXTRACT(YEAR FROM order_date);
```

- 주문 날짜에서 연도를 추출하여 연도별 주문 수를 계산합니다.

16. 연도별, 분기별 주문 수 출력하기

```
SELECT EXTRACT(YEAR FROM order_date) AS order_year
      ,EXTRACT(QUARTER FROM order_date) AS quarter
      ,COUNT(*) AS order_count
FROM orders
GROUP BY EXTRACT(YEAR FROM order_date), EXTRACT(QUARTER FROM order_date)
ORDER BY EXTRACT(YEAR FROM order_date), EXTRACT(QUARTER FROM order_date);
```

- 연도별, 분기별로 주문 수를 계산하고, 연도와 분기를 기준으로 정렬합니다.

17. 월별로 주문 수 출력하기

```
SELECT EXTRACT(MONTH FROM order_date) AS order_month
      ,COUNT(*) AS order_count
FROM orders
WHERE required_date < shipped_date
GROUP BY EXTRACT(MONTH FROM order_date)
ORDER BY EXTRACT(MONTH FROM order_date);
```

- 주문이 출하일보다 늦게 요구된 경우에 대해서만 월별 주문 수를 계산하고, 월별로 정렬합니다.

18. 특정 제품의 재고 개수 구하기

```
SELECT product_name
      ,SUM(stock) AS total_stock
FROM product
WHERE product_name LIKE '%우유%'
GROUP BY product_name;
```

- 제품 이름에 '우유'가 포함된 제품들의 재고를 합산하여 출력합니다.

19. VIP 고객과 일반 고객 구분하여 회원 수 및 평균 마일리지 구하기

```
SELECT CASE
           WHEN mileage >= 100 THEN 'VIP고객'
           ELSE '일반고객'
       END AS customer_type
      ,COUNT(*) AS customer_count
```

```
      ,AVG(mileage) AS average_mileage
FROM customer
GROUP BY CASE
                WHEN mileage >= 100 THEN 'VIP고객'
                ELSE '일반고객'
            END;
```

- 마일리지가 100 이상이면 VIP 고객으로 분류하고, 그렇지 않으면 일반 고객으로 분류합니다. 고객 유형별로 회원 수와 평균 마일리지를 계산합니다.