PostgreSQL Aggregations & Window Functions – Complete Structured Notes

---

1. What are Aggregations?

Aggregation functions **summarize multiple rows into a single value**. They are commonly used in reporting, analytics, and data engineering.

---

2. Sample Table Used in Examples

```
CREATE TABLE employe (
  emp_id INT,
  emp_name VARCHAR,
  dep VARCHAR,
  salary INT
);
```

```
INSERT INTO employe VALUES
(1,'E1','D1',20000),
(2,'E2','D1',30000),
(3,'E3','D1',40000),
(4,'E4','D2',15000),
(5,'E5','D2',25000),
(6,'E6','D2',35000),
(7,'E7','D2',35000);
```

Data in employe table

emp_id	emp_name	dep	salary
1	E1	D1	20000
2	E2	D1	30000
3	E3	D1	40000
4	E4	D2	15000
5	E5	D2	25000
6	E6	D2	35000
7	E7	D2	35000

3. Simple Aggregation Functions

3.1 SUM()

Returns **total of a numeric column**.

```
SELECT SUM(salary) FROM employe;
```

3.2 MIN()

Returns **minimum value**.

```
SELECT MIN(salary) FROM employe;
```

3.3 MAX()

Returns **maximum value**.

```
SELECT MAX(salary) FROM employe;
```

3.4 AVG()

Returns **average value**.

```
SELECT AVG(salary) FROM employe;
```

3.5 COUNT()

Returns **number of non-null rows**.

```
SELECT COUNT(salary) FROM employe;
```

---

4. GROUP BY Aggregations

Used to apply aggregation **per group**.

Rule

> Any column in SELECT that is not aggregated must appear in GROUP BY.

4.1 SUM per Department

```sql
SELECT dep, SUM(salary)
FROM employe
GROUP BY dep;
```

4.2 MIN per Department

```sql
SELECT dep, MIN(salary)
FROM employe
GROUP BY dep;
```

4.3 MAX per Department

```sql
SELECT dep, MAX(salary)
FROM employe
GROUP BY dep;
```

4.4 AVG per Department

```sql
SELECT dep, AVG(salary)
FROM employe
GROUP BY dep;
```

4.5 COUNT per Department

```sql
SELECT dep, COUNT(salary)
FROM employe
GROUP BY dep;
```

---

5. Window Functions (Ranking Functions)

Window functions **do not collapse rows**. They return a value for **each row** while still using aggregation logic.

---

6. Ranking Concepts Explained

Salary Values Example

```
10, 20, 20, 30
```

Function	Output
ROW_NUMBER	1 2 3 4
RANK	1 2 2 4
DENSE_RANK	1 2 2 3

---

7. ROW_NUMBER, RANK, DENSE_RANK (Overall)

```sql
SELECT emp_name, salary,
ROW_NUMBER() OVER (ORDER BY salary DESC) AS rn,
RANK() OVER (ORDER BY salary DESC) AS rk,
DENSE_RANK() OVER (ORDER BY salary DESC) AS dr
FROM employe;
```

---

8. Ranking with PARTITION BY (Department-wise)

```sql
SELECT emp_name, dep, salary,
ROW_NUMBER() OVER (PARTITION BY dep ORDER BY salary DESC) AS rn,
RANK() OVER (PARTITION BY dep ORDER BY salary DESC) AS rk,
DENSE_RANK() OVER (PARTITION BY dep ORDER BY salary DESC) AS dr
FROM employe;
```

---

9. Filtering Ranked Results (Second Highest Salary per Department)

Step 1: Create ranking

```
SELECT emp_name, dep, salary,
DENSE_RANK() OVER (PARTITION BY dep ORDER BY salary DESC) AS dr
FROM employe;
```

Step 2: Filter using subquery

```
SELECT * FROM (
  SELECT emp_name, dep, salary,
  DENSE_RANK() OVER (PARTITION BY dep ORDER BY salary DESC) AS dr
  FROM employe
) t
WHERE dr = 2;
```

---

10. Important Differences (Exam Favorite)

Function	Skips Numbers	Handles Ties
ROW_NUMBER	No	No
RANK	Yes	Yes
DENSE_RANK	No	Yes

---

11. GROUP BY vs WINDOW FUNCTIONS

Feature	GROUP BY	WINDOW FUNCTION
Rows reduced	Yes	No
Used for totals	Yes	Yes
Ranking possible	No	Yes
Per-row output	No	Yes

---

12. Exam One-Line Definitions

- **SUM**: Returns total of values
- **AVG**: Returns average value

- **COUNT**: Returns number of rows
- **GROUP BY**: Groups rows for aggregation
- **ROW_NUMBER**: Unique sequence per row
- **RANK**: Same rank for ties, skips next
- **DENSE_RANK**: Same rank for ties, no gaps

✅These notes are **Canva-ready**, **exam-oriented**, and **interview-safe**.

If you want next: - HAVING clause notes - Real-time salary problems - PostgreSQL interview questions - Aggregations with JOINs

Just tell me ♀