

Aggregate Functions

CS 2550 Database Design 2023
Cody Henrichsen

1

Definition

- Aggregate functions operate on an entire COLUMN's worth of values compared to the values in individual cells
- The column type must 'fit' the function
 - You can MIN a date or text
 - You can't AVG a text or a date

id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct
1	Jeffrey	Turner	JT	916 555 1234	1980-09-01	SA_REP	3100	0.1
2	Wendell	Wong	WW	916 555 2345	1985-01-01	SA_REP	2500	0.1
3	John	King	JK	916 555 3456	1981-02-01	SA_REP	2100	0.1
4	Deena	Deiwick	DD	916 555 4567	1986-03-01	SA_REP	2900	0.1
5	Alexis	Baer	AB	916 555 5678	1986-04-01	SA_REP	3200	0.1
6	Julia	Abadi	JA	916 555 6789	1987-05-01	SA_REP	2600	0.1
7	Keith	Johnson	KJ	916 555 7890	1986-06-01	SA_REP	2800	0.1
8	Greg	King	GK	916 555 8901	1986-07-01	SA_REP	2900	0.1
9	Cheryl	Carson	CC	916 555 9012	1986-08-01	SA_REP	3000	0.1
10	David	Lee	DL	916 555 9123	1986-09-01	SA_REP	3100	0.1
11	Neena	Kochhar	NK	916 555 9234	1987-10-01	SA_REP	3200	0.1
12	Lex	DeHaan	LD	916 555 9345	1987-11-01	SA_REP	3300	0.1
13	Victoria	Johnson	VJ	916 555 9456	1988-12-01	SA_REP	3400	0.1
14	Martin	Smith	MS	916 555 9567	1988-01-01	SA_REP	3500	0.1
15	Anthony	Pavlow	AP	916 555 9678	1988-02-01	SA_REP	3600	0.1
16	Pat	Haas	PH	916 555 9789	1988-03-01	SA_REP	3700	0.1
17	Shelley	Stevens	SS	916 555 9890	1988-04-01	SA_REP	3800	0.1
18	Timothy	Gietz	TG	916 555 9901	1988-05-01	SA_REP	3900	0.1
19	Lisa	Wu	LW	916 555 9012	1988-06-01	SA_REP	4000	0.1
20	Den	Raph	DR	916 555 9123	1988-07-01	SA_REP	4100	0.1
21	Jan	Baer	JB	916 555 9234	1988-08-01	SA_REP	4200	0.1
22	Peter	Brunt	PB	916 555 9345	1988-09-01	SA_REP	4300	0.1
23	Samuel	McCoy	SM	916 555 9456	1988-10-01	SA_REP	4400	0.1
24	Michael	Green	MG	916 555 9567	1988-11-01	SA_REP	4500	0.1
25	Patricia	Winters	PW	916 555 9678	1988-12-01	SA_REP	4600	0.1
26	Henry	Ford	HF	916 555 9789	1989-01-01	SA_REP	4700	0.1
27	Elizabeth	Green	EG	916 555 9890	1989-02-01	SA_REP	4800	0.1
28	Vern	Green	VG	916 555 9901	1989-03-01	SA_REP	4900	0.1
29	Rene	Green	RG	916 555 9012	1989-04-01	SA_REP	5000	0.1
30	Kevin	Green	KG	916 555 9123	1989-05-01	SA_REP	5100	0.1
31	Tim	Gietz	TG	916 555 9234	1989-06-01	SA_REP	5200	0.1
32	John	Abadi	JA	916 555 9345	1989-07-01	SA_REP	5300	0.1
33	Christina	Baer	CB	916 555 9456	1989-08-01	SA_REP	5400	0.1
34	Greg	Green	GG	916 555 9567	1989-09-01	SA_REP	5500	0.1
35	Anthony	Green	AG	916 555 9678	1989-10-01	SA_REP	5600	0.1
36	Pat	Green	PG	916 555 9789	1989-11-01	SA_REP	5700	0.1
37	Shelley	Green	SG	916 555 9890	1989-12-01	SA_REP	5800	0.1
38	Timothy	Green	TG	916 555 9901	1990-01-01	SA_REP	5900	0.1
39	Lisa	Green	LG	916 555 9012	1990-02-01	SA_REP	6000	0.1
40	Den	Green	DG	916 555 9123	1990-03-01	SA_REP	6100	0.1
41	Jan	Green	JG	916 555 9234	1990-04-01	SA_REP	6200	0.1
42	Peter	Green	PG	916 555 9345	1990-05-01	SA_REP	6300	0.1
43	Samuel	Green	SG	916 555 9456	1990-06-01	SA_REP	6400	0.1
44	Michael	Green	MG	916 555 9567	1990-07-01	SA_REP	6500	0.1
45	Patricia	Green	PG	916 555 9678	1990-08-01	SA_REP	6600	0.1
46	Henry	Green	HG	916 555 9789	1990-09-01	SA_REP	6700	0.1
47	Elizabeth	Green	EG	916 555 9890	1990-10-01	SA_REP	6800	0.1
48	Vern	Green	VG	916 555 9901	1990-11-01	SA_REP	6900	0.1
49	Rene	Green	RG	916 555 9012	1990-12-01	SA_REP	7000	0.1
50	Kevin	Green	KG	916 555 9123	1991-01-01	SA_REP	7100	0.1

2

Which functions?

- COUNT
 - The number of rows that match
- AVG
 - The mean average numeric value
- MIN
 - The smallest value specified
- MAX
 - The largest value specified
- SUM
 - The total of all values in the column

3

How?

- ▶ All the aggregate functions take a single parameter and return a single value
- ▶ AVG/MAX/MIN/SUM each take the column, the results of which are analyzed or operated on.
- ▶ COUNT takes **ONLY** the * as its parameter as it returns the number of rows in the the query that are returned
 - ▶ On its own just rows in table
 - ▶ Especially useful with the GROUP BY clause for group counts

4

Extra Clauses

GROUP BY

- ▶ **REQUIRED** for ALL columns not linked to an aggregate function
- ▶ Columns separated by commas
- ▶ Extra columns invalidate results
- ▶ Order is not required but follow the SELECT clause
- ▶ Written after the FROM/WHERE clauses
- ▶ Executes **BEFORE** the SELECT clause

HAVING

- ▶ Optional
- ▶ Used to limit rows based on results of an aggregate function (like WHERE)
- ▶ Written after the GROUP BY clause
- ▶ Executes **before** the SELECT clause

5

When?

- ▶ If more columns than what the aggregate function is called on, the required GROUP BY clause is going to execute **AFTER** the FROM (of course) and WHERE clauses, but **BEFORE** the SELECT clause since the coalesced column values need to be grouped together (lol)
- ▶ The optional HAVING clause also occurs **BEFORE** the SELECT clause since this is how the now coalesced results are then filtered before becoming available to the SELECT clause

6

Execution order

1. FROM
 - Pick the table(s)
2. WHERE
 - Limit results based on other value(s)
3. GROUP BY
 - Identify and isolate based on column values
4. HAVING
 - Limit results based on result of aggregate
5. SELECT
 - Pick the columns to display

7

Execution order of a query

```
4 SELECT
  section_id,
  COUNT(*) AS enroll_count
1 FROM
  enrollment
2 GROUP BY
  section_id
3 HAVING
  COUNT(*) > 4
5 ORDER BY
  COUNT(*),
  section_id
;
```

8

Demo Query

Worksheet Query Builder

```
1 SELECT
2   section_id
3 FROM
4   enrollment;
```

Results

SECTION_ID
99
101
99
101
99
101

226 rows selected.

9

Limit query with DISTINCT

Distinct Query

```
6 SELECT DISTINCT
7   section_id
8 FROM
9   enrollment;
10
```

Distinct Results

SECTION_ID
102
100
147
117
123
92
150
138
91

64 rows selected.

10

Use Aggregate Function

Aggregate Query

```
11 SELECT
12   section_id,
13   COUNT(*) AS enroll_count
14 FROM
15   enrollment
16 GROUP BY
17   section_id;
18
```

Aggregate Results

SECTION_ID	ENROLL_COUNT
148	5
149	1
150	3
151	2
152	4
153	3
154	4
155	5
156	8

64 rows selected.

11
