

Aggregate operations on Relations

Given a relation, we perform some operations over all tuples or grouped tuples

ename character v	ssn integer	bda dat	gender charact	salary numerik	supers intege	dno sma
James	105	192	M	55000		1
Franklin	102	194	M	40000	105	5
Jennifer	106	193	F	43000	105	4
John	101	195	M	30000	102	5
Alicia	108	195	F	25000	106	4
Ramesh	104	195	M	38000	102	5
Joyce	103	196	F	25000	102	5
Ahmad	107	195	M	25000	106	4

Aggregation operations are basically counting or summing of an attribute-values. Following are common aggregation operations – COUNT, SUM, AVG, MAX, MIN for columns of a table.

Aggregation can be computed over all tuples or on grouped tuples of relations.

Following are examples of queries that require performing aggregate operations, and thus are called aggregate queries -

- Find out total salary we pay to all employees.
- Find out total number of employees, total number of supervisors, and so
- Find out department wise salary statistics, i.e., sum, avg, max, min
- Find out employee who are drawing less than average salary,
- Find out average salary paid to managers,
- And so forth

Aggregation operations are expressed using script F (F) operator

$\mathcal{F}_{\langle \text{function-list} \rangle}(\mathbf{r})$

Examples

$\mathcal{F}_{\text{COUNT(SSN), AVG(SALARY)}}(\text{employee})$

Written in SQL as

`SELECT count(ssn), avg(salary) FROM employee;`

The result of this operation will be a single tuple having two columns.

Another example

$\mathcal{F}_{\text{COUNT(SSN), MAX(SALARY), MIN(SALARY), AVG(SALARY)}}(\text{employee})$

`SELECT count(ssn), max(salary), min(salary), avg(salary) FROM employee;`

Aggregation over grouped tuples

In this case tuples of operand relation are grouped based on some attributes(s) value(s). We call these attributes as grouping attributes.

For every distinct value of grouping attribute(s), there will be a group, and then aggregation operation is computed for each group.

$$\langle \text{grouping-attributes} \rangle \mathcal{F} \langle \text{function-list} \rangle (\mathbf{r})$$

Example:

DNO \mathcal{F} COUNT(SSN), AVG(SALARY) (EMPLOYEE)


ename character v	ssn integer	bda date	gender character	salary numeric	supers integer	dno smallint
James	105	192	M	55000		1
Franklin	102	194	M	40000	105	5
Jennifer	106	193	F	43000	105	4
John	101	195	M	30000	102	5
Alicia	108	195	F	25000	106	4
Ramesh	104	195	M	38000	102	5
Joyce	103	196	F	25000	102	5
Ahmad	107	195	M	25000	106	4

DNO \mathcal{F} COUNT(SSN), AVG(SALARY) (EMPLOYEE)

ename	ssn	bda	gender	salary	supers	dno
character v	integer	date	charact	numeric	integer	small
James	105	192	M	55000		1

dno	count	avg
sma	bigint	numeric
1	1	55000.0

eno	count	avg
sm	bigint	numeric
4	3	31000.0



ename	ssn	bda	gender	salary	supers	dn
character v	integer	dat	charact	numeric	intege	sm
Jennifer	106	193	F	43000	105	4
Alicia	108	195	F	25000	106	4
Ahmad	107	195	M	25000	106	4

dno sma	count bigint	avg numeric
4	3	31000.0

ename character v	ssn integer	bda date	gender character	salary numeric	supers integer	dno smallint
Franklin	102	194	M	40000	105	5
John	101	195	M	30000	102	5
Ramesh	104	195	M	38000	102	5
Joyce	103	196	F	25000	102	5

dno	count	avg
sma	bigint	numeric
5	4	33250.0

dno	count	avg
sma	bigint	numeric
5	4	33250.0
4	3	31000.0
1	1	55000.0

```
SELECT dno, count(ssn), avg(salary) FROM employee GROUP BY dno;
```

Another example

DNO, GENDER \mathcal{F} COUNT(SSN), AVG(SALARY) (EMPLOYEE)

```
SELECT dno, gender, count(ssn), avg(salary)
FROM employee GROUP BY dno, gender;
```

Renaming of operations in aggregation

You can have renaming used in aggregate operation,

For example, an aggregate query-

$DNO \not\rightarrow COUNT(SSN) \rightarrow No_of_Emps, AVG(SALARY) \rightarrow AVG_SAL (EMPLOYEE)$

In SQL, we write as following -

```
SELECT dno, count(ssn) AS No_of_Emps, avg(salary) AS avg_sal
FROM employee
GROUP BY dno;
```

NULL's in Aggregation

- NULL never contribute to sum, average, or count, and can never be the minimum or maximum of a column.
- But if there are no non-NULL values in a column, then the result of the aggregation is NULL.

Examples

```
SELECT count(*) FROM employee; -- counts all row, * is used for counting rows!
SELECT count(dno) FROM employee; -- counts occurrence of dno values (excludes NULL)
SELECT count(DISTINCT dno) FROM employee; -- counts of distinct values for dno attribute
SELECT count(superssn) FROM employee;
SELECT count(DISTINCT superssn) FROM employee;
SELECT min(salary), max(salary), avg(salary) FROM employee;
```

Exercise: Can you figure out, why following references (in red) are invalid?

```
SELECT dno, ssn, avg(salary) AS avg_sal FROM employee GROUP BY dno;
```

```
SELECT dno, avg(salary) AS avg_sal FROM employee;
```

HAVING clause

HAVING is used to specify restrict over result of aggregation

For example:

```
SELECT dno, avg(salary) AS avg_sal
FROM employee GROUP BY dno
HAVING avg(salary) > 50000;
```

Algebraically, equivalent to

```
r1 ← DNO ⋈ AVG(SALARY) → AVG_SAL (EMPLOYEE)
result ← σavg(sal) > 50000 (r1)
```

Note that you cannot use avg_sal instead of avg(salary) in HAVING clause, because renaming is done at the time of projection.

Semantics of SQL SELECT statement

```
SELECT <attrib and/or function-list> (5)
FROM <relation-expression> (1)
[WHERE <condition>] (2)
[GROUP BY <grouping attribute(s)>] (3)
[HAVING <group-filter-condition>] (4)
[ORDER BY <attrib-list>]; (6)
```

```
r1 ← <relational-expression>
r2 ← σ<where-condition>(r1)
r3 ← <group-attributes> F<aggregate-operation>(r2)
r4 ← σ<group-filter-condition>(r3)
r5 ← π<attrib-list>(r4)
```

- Result of FROM and WHERE is given to GROUP BY operation
- GROUP BY operation computes *aggregated values* for each group value, and gives you one tuple for each group value.
- **group-filter-conditions** in HAVING are used to apply restriction over result of GROUP BY operation
- Finally project is applied as defined in SELECT clause of SELECT statement.