



**Database Design and Development
Implementation – Aggregate Functions &
Grouping**

Name: _____

Contents

Aggregate functions	3
Worked Examples – Aggregate Functions	6
Worked Examples - Grouping	8
Practical Task – Aggregate Functions.....	11

Aggregate functions

An Aggregate function is a function that groups together the values of multiple rows to return a single statistical value.

The most common aggregate functions used are listed below:

Function	Description
AVG ()	returns the average value of a numeric column or expression
COUNT ()	returns the number of rows that match the criteria in the WHERE clause
MAX ()	returns the largest value of the selected column or expression
MIN ()	returns the smallest value of the selected column or expression
SUM ()	returns the total sum of a numeric column or expression

In the same way that pre-defined programming functions receive parameter values, SQL aggregate functions require an expression. This expression is usually a column name but it can be a column name together with an operator.

SUM() and AVG() can only be applied to numeric data types

MIN() and MAX() work with characters, numeric, and date/time datatypes;

COUNT() works with all data types.

All aggregate functions except, COUNT(), ignore nulls.

COUNT() always returns a positive integer or zero. The other aggregate functions return null if the set contains no rows or contains rows with only nulls.

An aggregate expression cannot be used in a WHERE clause.

Use of more than one aggregate expression

It is possible to use more than one aggregate expression in a SELECT statement as shown here:

```
SELECT MIN(price), MAX(price)
FROM Product;
```

Incorrect SELECT Statement

Mixing non-aggregate and aggregate expressions in a SELECT statement is not permitted. A SELECT statement must contain either all non-aggregate expressions or all aggregate expressions. The query below is illegal, as it mixes non-aggregate productName with the aggregate function MAX.

```
SELECT productName, MAX(price)
FROM Product;
```



Readable Headings

Example: Use of readable headings to display the cheapest and dearest price per night.

```
SELECT MIN(pricePersonNight) AS [Cheapest Price per Night],
MAX(pricePersonNight) AS [Dearest Price perNight]
FROM Hotel;
```

Count

Used to display a list of the different types of resort together with the number of resorts in each of those categories

```
SELECT resortType, COUNT(*)  
  
FROM Resort  
  
GROUP BY resortType;
```

Grouping (group by) gathers together the rows of a table or answer table based on a column or columns with the same value or values.

SUM

Uses a readable heading to display the total number of people booked into a hotel in July

```
SELECT SUM(numberInParty) AS [People on holiday in July]  
  
FROM Booking  
  
WHERE startDate LIKE '*/07/*';
```

Worked Examples – Aggregate Functions

Design a query to display the shortest and longest assessment durations.

Field(s) and calculation(s)	MIN(duration), MAX(duration),
Table(s) and query	Assessment
Search criteria	
Grouping	
Sort Order	

```
SELECT min(duration), max(duration)
FROM assessment;
```

Design a query that uses *readable* headings to display the longest and shortest assessment duration.

Field(s) and calculation(s)	Longest = MIN(duration), Shortest = MAX(duration)
Table(s) and query	Assessment
Search criteria	
Grouping	
Sort Order	

```
SELECT min(duration) AS shortest, max(duration) AS
longest
FROM assessment;
```

Design a query that uses *readable* headings to display the average and total assessment durations.

Field(s) and calculation(s)	Average duration= AVG(duration), Total of Pass Marks = SUM(duration)
Table(s) and query	Assessment
Search criteria	
Grouping	
Sort Order	

```
SELECT avg(duration) AS [Average duration],
sum(passmark) AS [total of pass marks]

FROM assessment;
```

Design a query that uses a *readable* heading to display the number of assessments.

Field(s) and calculation(s)	Number of Assessments = COUNT(*)
Table(s) and query	Assessment
Search criteria	
Grouping	
Sort Order	

```
SELECT count(*) AS [Number of Assessments]

FROM assessment;
```

Worked Examples - Grouping

Design a query to display the different types of assessment.

Field(s) and calculation(s)	assessmentType
Table(s) and query	Assessment
Search criteria	
Grouping	assessmentType
Sort Order	

```
SELECT assessmentType
FROM assessment
GROUP BY assessmentType;
```

Design a query to display the different types of assessment, together with the number of assessments in those categories

Field(s) and calculation(s)	assessmentType, COUNT(*)
Table(s) and query	assessment
Search criteria	
Grouping	assessmentType
Sort Order	

```
SELECT assessmentType, count(*) AS [number of
assessments]
FROM assessment
GROUP BY assessmentType;
```


Design a query to display the different types of assessment and their writers, together with the number of assessments in those categories

Field(s) and calculation(s)	assessmentType, writer, COUNT(*)
Table(s) and query	assessment
Search criteria	
Grouping	assessmentType, writer
Sort Order	

```
SELECT assessmentType, writer, count(*) AS [number of
assessments]

FROM assessment

GROUP BY writer, assessmenType;
```

Design a query to display the different types of assessment, together with the longest assessment and the lowest pass mark in each category.

Field(s) and calculation(s)	assessmentType, Longest Assessment = MAX(duration), Lowest Pass Mark = min(passmark)
Table(s) and query	assessment
Search criteria	
Grouping	assessmentType
Sort Order	

```
SELECT assessmenttype, max(duration) AS [Longest
Assessment], min(passmark) AS [Lowest Pass Mark]

FROM assessment

GROUP BY assessmenttype;
```

Design a query to display the different types of assessment, except practical assessments, together with the longest assessment in each category.

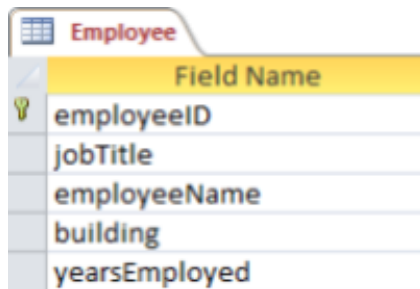
Field(s) and calculation(s)	assessmentType, Longest Assessment = MAX(duration), Lowest Pass Mark = min(passmark)
Table(s) and query	assessment
Search criteria	
Grouping	assessmentType
Sort Order	

```
SELECT assessmenttype, max(duration) AS [Longest  
Assessment]  
  
FROM assessment  
  
WHERE assessmenttype <>"Practical"  
  
GROUP BY assessmenttype;
```

Practical Task – Aggregate Functions

Task 1

Open the file called Employees Database. This database has one table called Employee.



The image shows a screenshot of a database tool displaying the structure of the 'Employee' table. The table has a yellow header row with the text 'Field Name'. Below the header, there are five rows, each representing a field in the table. The fields are: 'employeeID', 'jobTitle', 'employeeName', 'building', and 'yearsEmployed'. A small lightbulb icon is visible to the left of the 'employeeID' field.

Field Name
employeeID
jobTitle
employeeName
building
yearsEmployed

Use SQL queries to display each set of required details.

1. List the longest and shortest service of all employees of the company.
2. List the average years of service of all the employees.
3. List the job titles and the average years of service for each of those job titles.
4. List the name of each building and the number of employees who work in it.
5. List the name of each building and the total years service of all employees who work in that building.
6. Employees in each building are due a bonus of 100 times the shortest years of service of any employee who works in that building.

List the name of each building and the bonus due to each employee in the building.

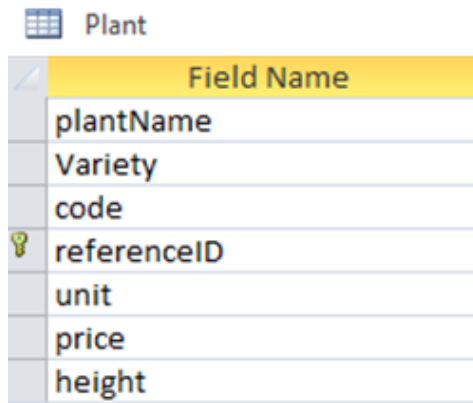
7. All of the employees with the same job title are due to receive extra holiday days equivalent to half of the longest years of service for that job title.

List each job title together with the extra holiday days due for those employees.

8. List the name of each building with the job titles of the employees who work in the building and the number of employees with these job titles.

Task 2

Open the file called Plants Database. This database has one table called Plant.



	Field Name
	plantName
	Variety
	code
🔑	referenceID
	unit
	price
	height

Use SQL queries to display each set of required details.

1. Display the name of each category and the average price of plants in that category (round the average price to 2 decimal places).
2. Display the plant codes together with the number of plants that share the same code. The code with the most plants should be listed first; codes that have the same number of plants must be listed in alphabetical order of code.
3. Display the list of plant heights and the range of the prices for each plant height (the range of prices is the difference between the dearest and the cheapest plant).
4. Display the number of plants that have the letter 'p' in their name together with the total cost of those plants.
5. Display the largest and average unit size of all plants that have a referenceID beginning with the letter 'B'.
6. Display the list of unit sizes together with the number of plants and the cheapest plant sold in those units. The most populous unit size should be listed first; where the number of plants is the same, the plants should be listed from smallest to largest unit size.
7. Display the list of categories and heights together with the average price of the plants in each sub-category of height.