Topic C

SQL QUERY 2 (Row-wise/Aggregate Functions/Summarizing Results By Group)

SQL Query 2 (Row-wise/Aggregate Functions/Summarizing Results By Group)

CONTENT

- Row-wise operations
- Aggregate functions
- Summarizing results by group

Row-wise operation

Product_Code	Selling_price	Prod_Description	Accessory _price
HG5000	17.60	Body Pants + bell	2.40
RQ8996	22.00	Dress Yellow	NULL
DT4000	17.20	Top + Skirt White + necklace	12.80
RQ7601	26.50	Dress Colour	NULL

Product

What is the total selling price for each product?

How much will it cost if I buy 5 of each product?

Original cost price table			Total selling price per product	Total selling price for 5
Product_code	Selling_price	Accessory_price	Selling_price + accessory_pice	(Selling_price + accessory_price) * 5
HG5000	17.60	2.40	20	100
RQ8996	22	NULL	NULL	NULL
DT4000	17.20	12.80	30	150
RQ7601	26.50	NULL	NULL	NULL

Row-wise operation

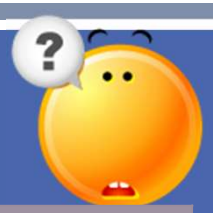
Orignal cost price table			Total selling price per item	Total selling price for 5 items
Product_code	Selling_ price	Accessory_price	Selling_price + accessory_price	(Selling_price + accessory_price) * 5
HG5000	17.60	2.40	20	100
RQ8996	22	NULL	NULL	NULL
DT4000	17.20	12.80	30	150
RQ7601	26.50	NULL	NULL	NULL

What is the total selling price for each product?

How much will it cost if I buy 5 of each product?

```
SQL
SELECT
Product_code  'Product code',    Prod_Description 'Product description' ,
Selling_price  'Unit price',     Accessory_price  'Accessory price',
Selling_price + accessory_price    'Total selling price',
(Selling_price + accessory_price) * 5   'Total selling price for 5'
FROM Product
```

Row-wise operation




```
SELECT
product_code  'Product code',   Prod_Description  'Product description' ,
Selling_price 'Selling price',  accessory_price   'Accessory price',
Selling_price + accessory_price    'Total selling price',
(Selling_price + accessory_price) * 5        'Total selling price for 5'
FROM Product
```

Product code	Product description	Selling price	Accessory price	Total selling price	Total selling price for 5 items
HG5000	Body pants + bell	17.60	2.40	20	100
RQ8996	Dress yellow	22	NULL	NULL	NULL
DT4000	Top + White dress + necklace	17.20	12.80	30	150
RQ7601	Dress colour	26.50	NULL	NULL	NULL

Resulting Table

If one of the columns involve in a column expression is ***NULL***, the column expression returns a ***NULL***

Row-wise operations - ISNULL function

Product code	Product description	Selling price	Accessory price	Total selling price	Total selling price for 5 items
HG5000	Body pants + bell	17.60	2.40	20	100
RQ8996	Dress yellow	22	NULL	NULL	NULL
DT4000	Top + White dress + necklace	17.20	12.80	30	150
RQ7601	Dress colour	26.50	NULL	NULL	NULL

Resulting Table

How to resolve the NULL issue?

ISNULL (argument1, argument2)

If argument1 is NULL, then return argument2
If argument1 is not NULL, then return argument 1

Row-wise operation - ISNULL function

Product

Product code	Product description	Selling price	Accessory price	Total selling price	Total selling price for 5 items
HG5000	Body pants + bell	17.60	2.40	20	100
RQ8996	Dress yellow	22	NULL	NULL	NULL
DT4000	Top + White dress + necklace	17.20	12.80	30	150
RQ7601	Dress colour	26.50	NULL	NULL	NULL

SQL

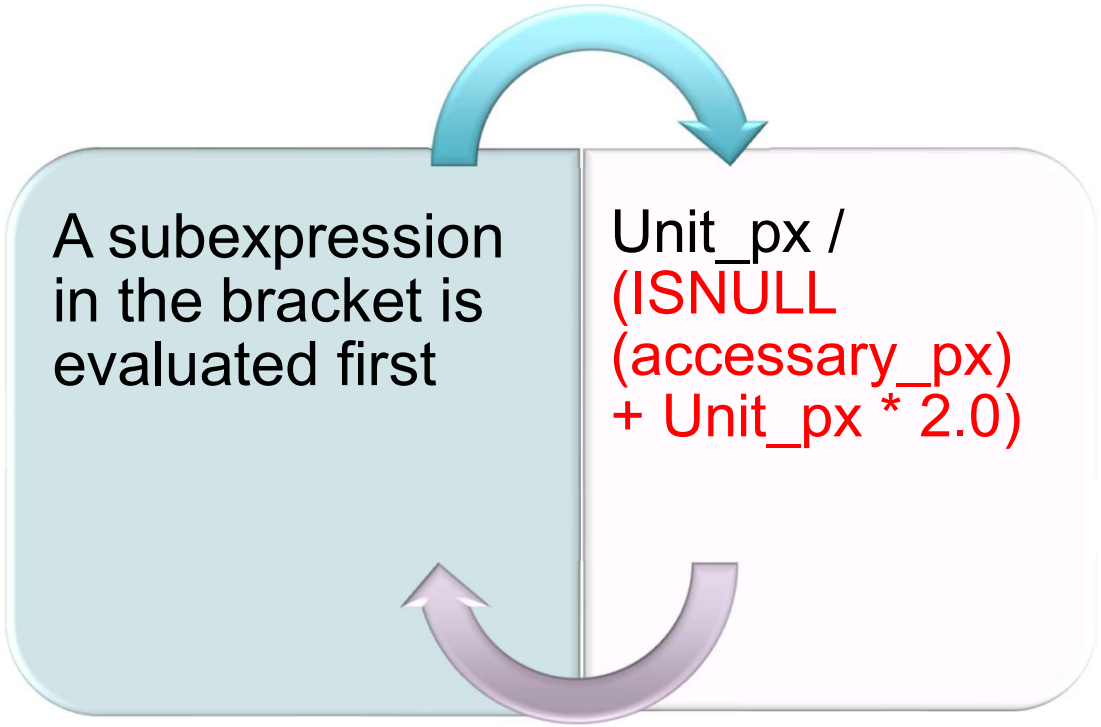
```
SELECT
product_code                    'Product code',     Prod_Description        'Product description' ,
Selling_price                   'Selling price'     accessory_price         'Accessory price',
ISNULL(selling_price, 0) + ISNULL(accessory_price, 0)                       'Total selling price',
(ISNULL(selling_price. 0) + ISNULL(accessory_price, 0)) * 5                 'Total selling price for 5'
FROM Product
```

ALL NULL REPLACE BY 0

Resulting Table

Product code	Product description	Selling price	Accessory price	Total selling price	Total selling price for 5 items
HG5000	Body pants + bell	17.60	2.40	20	100
RQ8996	Dress yellow	22	NULL	22	110
DT4000	Top + White dress + necklace	17.20	12.80	30	150
RQ7601	Dress colour	26.50	NULL	26.50	132.50

Row-wise operation - Using numeric operator in expression



C

Row-wise operation - Using string operator in expression

Product_Code	Selling_price	Prod_Description	Accessory _price
HG5000	17.60	Body Pants + bell	2.40
RQ8996	22.00	Dress Yellow	NULL
DT4000	17.20	Top + Skirt White + necklace	12.80
RQ7601	26.50	Dress Colour	NULL

Product

To display the product code with description by embedding a space between them. Example: RQ8996 Dress Yellow

SQL

```
SELECT product_code + ' ' + prod_description as Name_of_Product
FROM Product
```

The '+' sign represents the string operation known as concatenation.
The ' ' represents a space.

Name_of_Product
HG5000 Body pants + bell
RQ8996 Dress yellow
DT4000 Top + White dress + necklace
RQ7601 Dress colour

Resulting Table

C

Row-wise opeataion - Using built-in function – SUBSTRING()

Product_Code	Selling_price	Prod_Description	Accessory _price
HG5000	17.60	Body Pants + bell	2.40
RQ8996	22.00	Dress Yellow	NULL
DT4000	17.20	Top + Skirt White + necklace	12.80
RQ7601	26.50	Dress Colour	NULL

Product

How to **extract 3 characters** starting **from 2nd character** of the description? Example: ody from Body Pants + bell

Start from 2nd character

SQL

```
SELECT prod_description, SUBSTRING(prod_description, 2, 3) as 'Extract 3 characters from 2nd position'
FROM Product
```

Extract 3 characters

To extract part of the string.

Prod_Description	Extract 3 characters from 2nd positioin
Body Pants + bell	ody
Dress Yellow	res
Top + Skirt White + necklace	op
Dress Colour	res

Resulting Table

C

Row-wise operation - Using Day, Month, Year function to extract dd, mm, yyyy of a date attribute

Email	Password	DOB
LindaS@hotmail.com	Abc123	NULL
DavidL@gmail.com	Da12lee	13/09/1997
LSoh@hotmail.com	Abc123	20/12/1998

Customer

To list the year each customer was born.

```
SQL
SELECT email, dob, YEAR(dob) _Year_of_Birth
FROM Customer
```

To extract the year from the date.

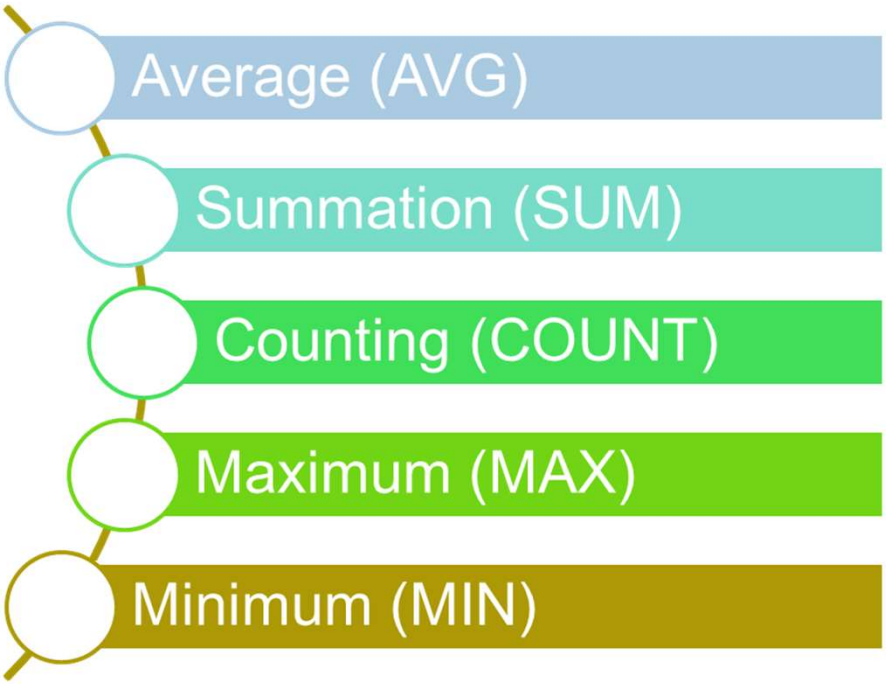
Email	DOB	Year_of_Birth
LindaS@hotmail.com	NULL	NULL
DavidL@gmail.com	13/09/1997	1997
LSoh@hotmail.com	20/12/1998	1998

Resulting Table

Row-wise operation
Summary of built-in functions

Function	Output
GETDATE()	Current date/time
UPPER(string)	Convert string to uppercase
LOWER(string)	Convert string to lowercase
SUBSTRING(Source, s, l)	Part of source starting at position s of length l
LEN(string)	The number of character of source
Cast(columnName as datatype(size))	To change datatype for a given column/expression
Convert(datatype(size), columnName, [style])	To change datatype for a given column/expresession

Aggregate functions



Operates on columns of data

Aggregate functions

	Column Expression (Argument to Aggregation Function)				
Aggregate Function	Product_Code	Description	Unit_px	Accessory _px	(Unit_px + Accessory_px) * 5
	HG5000	Body Pants + bell	10	5	75
	RQ8996	Dress Yellow	25	NULL	NULL
	DT4000	Top + Skirt White + necklace	15	10	125
	RQ7601	Dress Colour	30	NULL	NULL

Sum
Average
Max
Min
Count
Count(*)

Product

Aggregate functions - AVG (Column_Expression)

Product_Code	Description	Unit_px	Accessory _px
HG5000	Body Pants + bell	10	5
RQ8996	Dress Yellow	25	NULL
DT4000	Top + Skirt White + necklace	15	10
RQ7601	Dress Colour	30	NULL

Product

What is the average Accessory_px for the products ?

SQL
SELECT AVG(accessory_px) 'Average Accessory Price'
FROM Product

Average Accessory Price
7.5

Resulting Table

Average the NON-NULL values defined by the column expression

Aggregate functions - SUM (Column_Expression)

What is the sum of Accessory_px for the products ?

Product_Code	Description	Unit_px	Accessory _px
HG5000	Body Pants + bell	10	5
RQ8996	Dress Yellow	25	NULL
DT4000	Top + Skirt White + necklace	15	10
RQ7601	Dress Colour	30	NULL

Product

```
SELECT SUM(accessory_px)   'Total Accessory Price'
FROM Product
```

Total Accessory Price
15

Resulting Table

Total the NON-NULL values defined by the column expression

Aggregate functions - COUNT (Column_Expression)

How many products have Accessory_px ?

Product_Code	Description	Unit_px	Accessory _px
HG5000	Body Pants + bell	10	5
RQ8996	Dress Yellow	25	NULL
DT4000	Top + Skirt White + necklace	15	10
RQ7601	Dress Colour	30	NULL

Product

SQL

```
SELECT COUNT(accessory_px)   'Number of products with accessory price'
FROM Product
```

Number of products with accessory price
2

Resulting Table

Count the NON-NULL values defined by the column expression

Aggregate functions COUNT (Column_Expression)

Can I count the number of NULL?

Product_Code	Description	Unit_px	Accessory _px
HG5000	Body Pants + bell	10	NULL
RQ8996	Dress Yellow	25	NULL
DT4000	Top + Skirt White + necklace	15	NULL
RQ7601	Dress Colour	30	NULL

Product

SQL
SELECT COUNT(accessory_px) ‘Number of NULL’
FROM Product

Number of NULL
0

Resulting Table

Aggregate function-MAX (column_expression) & MIN (column_expression)

Product_Code	Description	Unit_px	Accessory _px
HG5000	Body Pants + bell	10	5
RQ8996	Dress Yellow	25	NULL
DT4000	Top + Skirt White + necklace	15	10
RQ7601	Dress Colour	30	NULL

Product

What is the highest and lowest accessory price?

```
SQL
SELECT MAX(accessory_px)   'Highest accessory price'
MIN(accessory_px) 'Lowest accessory price'
FROM Product
```

Highest accessory price	Lowest accessory price
10	5

Resulting Table

Return the HIGHEST/LOWEST NON-NULL values of the column expression of any data type (numeric, text or date)

Aggregate function - Usage of DISTINCT keyword in Aggregate functions

Product_Code	Description	Type_Of_product	Unit_px	Accessory _px
HG5000	Body Pants + bell	Pant	10	NULL
RQ8996	Dress Yellow	Dress	25	NULL
DT4000	Top + Skirt White + necklace	Top	15	NULL
RQ7601	Dress Colour	Dress	30	NULL

Product

SQL

```
SELECT COUNT(type_of_product) 'Number of product (Duplicates),
COUNT (DISTINCT type_of_product) 'Number of product (No Duplicates)
FROM Product
```

Number of Product (Duplicates)	Number of product (No Duplicates)
4	3

Resulting Table

If DISTINCT keyword is used in the column expression, then duplicate values of the column expression are ignored.

C

Aggregate function - Usage of DISTINCT keyword in Aggregate functions

Product_Code	Description	Type_Of_product	Unit_px	Accessory _px
HG5000	Body Pants + bell	Pant	45	5
RQ8996	Dress Yellow	Dress	25	NULL
DT4000	Top + Skirt White + necklace	Top	30	10
RQ7601	Dress Colour	Dress	25	NULL

Product

SQL
SELECT AVG(unit_px) 'Average unit price (Duplicates),
AVG (DISTINCT unit_px) 'Average unit price (No Duplicates)
FROM Product

Average unit price (Duplicates)	Average unit price (No Duplicates)
30	33.33

(45+25+30) / 3

(45+25+30+25) / 4

Resulting Table

Summarizing results by group - Using GROUP BY clause

What is the average price and quantity for each Supplier?

Stock

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG7166	Dress Blue	15.90	NULL	10
HG6159	Sale Dress Pink	15.40	S1002	40
HT5402	Pink Skirt	15.00	NULL	20

Summarizing results by group - Using GROUP BY clause

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG7166	Dress Blue	15.90	NULL	10
HG6159	Sale Dress Pink	15.40	S1002	40
HT5402	Pink Skirt	15.00	NULL	20

Stock

SQL	Syntax
SELECT Supplier_ID 'Supplier ID',	SELECT <List of Grouping columns>,
AVG(unit_px) 'Average Unit Price', AVG(qty) 'Average Quantity'	<Aggregate Function>
FROM stock	FROM <Table>
GROUP BY Supplier_ID	GROUP BY <List of Grouping Columns>

Grouping Attributes

Supplier ID	Average Unit Price	Average Quantity
NULL	15.45	15
S1001	17.85	25
S1002	17	40

What is the average price and quantity for each supplier?

Resulting Table

Summarizing results by group - Using GROUP BY clause for Group Summary

SQL
SELECT Supplier_ID 'Supplier ID',
AVG(unit_px) 'Average Unit Price', AVG(qty) 'Average Quantity'
FROM stock
GROUP BY Supplier_ID

Supplier ID	Average Unit Price	Average Quantity
NULL	15.45	15
S1001	17.85	25
S1002	17	40

Resulting Table

- GROUP BY queries uses a GROUP BY clause
- Group rows by specific column values to produce a single summary row for each group
- **NULL values** are grouped together as a **NULL group**

Summarizing results by group - Using HAVING clause for selecting group

What is the average price and quantity for each supplier?
Display those average quantity that is more than 20

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG7166	Dress Blue	15.90	NULL	10
HG6159	Sale Dress Pink	15.40	S1002	40
HT5402	Pink Skirt	15.00	NULL	20

Stock

SQL	Syntax
SELECT Supplier_ID 'Supplier ID',	SELECT <List of Grouping columns>,
AVG(unit_px) 'Average Unit Price', AVG(qty) 'Average quantity'	<Aggregate Function>
FROM stock	FROM <Table>
GROUP BY Supplier_ID	GROUP BY <List of Grouping Columns>

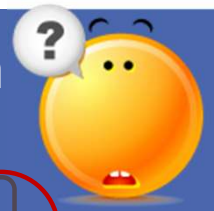
Supplier ID	Average Unit Price	Average Quantity
S1001	17.85	25
S1002	17	40

Resulting Table

Summarizing results by group - Using WHERE clause with GROUP BY

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG7166	Dress Blue	15.90	NULL	10
HG6159	Sale Dress Pink	15.40	S1002	40
HT5402	Pink Skirt	15.00	NULL	20

Stock



SQL	Syntax

So what is the result?

Summarizing results by group - Using WHERE clause with GROUP BY (Step 1)

Stock

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG7166	Dress Blue	15.90	NULL	10
HG6159	Sale Dress Pink	15.40	S1002	40
HT5402	Pink Skirt	15.00	NULL	20

WHERE Supplier_ID is not null

Intermediate Table 1

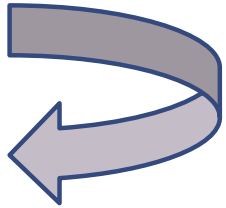
Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG6159	Sale Dress Pink	15.40	S1002	40

Summarizing results by group - Using WHERE clause with GROUP BY (Step 2)

Intermediate Table 1

GROUP BY Supplier_ID

Prod_Code	Prod_Desc	Unit_px	Supplier_ID	Qty
HG7160	Sale Dress White	15.90	S1001	30
HG9298	Sale Top + Skirt Red	19.80	S1001	20
RQ0207	Dress White	18.60	S1002	40
HG6159	Sale Dress Pink	15.40	S1002	40



Intermediate Table 2

Supplier_ID	Average Unit Price	Average Quantity
S1001	17.85	25
S1002	17	40

Summarizing results by group - Using WHERE clause with GROUP BY (Step 3)

Intermediate Table 2

Supplier_ID	Average Unit Price	Average Quantity
S1002	17	40
S1001	17.85	25

HAVING Avg(Qty) > 20

Intermediate Table 3

Supplier_ID	Average Unit Price	Average Quantity
S1001	17.85	25
S1002	17	40

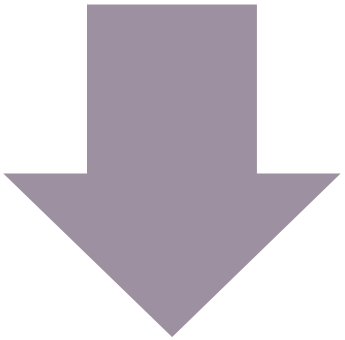
ORDER BY Supplier_ID DESC

Result table

Supplier_ID	Average Unit Price	Average Quantity
S1002	17	40
S1001	17.85	25

Summarizing results by group - Difference between WHERE and HAVING clauses

WHERE clause is executed first to filter rows before grouping them



HAVING clause filters groups for display