Department of
Computer Science and Engineering

Title: Implementation of MySQL Aggregate Function

Database System Lab

CSE 210

Green University of Bangladesh

1 Objective(s)

- Gather knowledge about the aggregate function.
- Implement different types of aggregate functions AVG, COUNT, SUM, MIN, MAX, UCASE, LCASE, FLOOR etc.

2 Problem analysis

We mainly use the aggregate functions in databases, spreadsheets and many other data manipulation software packages. In the context of business, different organization levels need different information such as top levels managers interested in knowing whole figures and not the individual details. These functions produce the summarised data from our database. Thus they are extensively used in economics and finance to represent the economic health or stock and sector performance.

MySQL aggregate functions	
Aggregate function	Description
AVG()	Return the average of non-NULL values.
BIT_AND()	Return bitwise AND.
BIT_OR()	Return bitwise OR.
BIT_XOR()	Return bitwise XOR.
COUNT()	Return the number of rows in a group, including rows with NULL values.
GROUP_CONCAT()	Return a concatenated string.
JSON_ARRAYAGG()	Return result set as a single JSON array.
JSON_OBJECTAGG()	Return result set as a single JSON object.
MAX()	Return the highest value (maximum) in a set of non-NULL values.
MIN()	Return the lowest value (minimum) in a set of non-NULL values.
STDEV()	Return the population standard deviation.
STDDEV_POP()	Return the population standard deviation.
STDDEV_SAMP()	Return the sample standard deviation.
SUM()	Return the summation of all non-NULL values a set.
VAR_POP()	Return the population standard variance.
VARP_SAM()	Return the sample variance.
VARIANCE()	Return the population standard variance.

SQL functions are similar to SQL operators in that both manipulate data items and both return a result. SQL functions differ from SQL operators in the format in which they appear with their arguments. The SQL function format enables functions to operate with zero, one, or more arguments.

function(argument1, argument2, ...) alias

If passed an argument whose datatype differs from an expected datatype, most functions perform an implicit datatype conversion on the argument before execution. If passed a null value, most functions return a null value.

SQL functions are used exclusively with SQL commands within SQL statements. There are two general types of SQL functions: single row (or scalar) functions and aggregate functions. These two types differ in the number of database rows on which they act. A single row function returns a value based on a single row in a query, whereas an aggregate function returns a value based on all the rows in a query.

Single row SQL functions can appear in select lists (except in SELECT statements that contain a GROUP BY clause) and WHERE clauses.

Aggregate functions are the set functions: AVG, MIN, MAX, SUM, and COUNT. You must provide them with an alias that can be used by the GROUP BY function.

2.1 Using Mathematical Function

Relational databases store information in tables — with columns that are analogous to elements in a data structure and rows which are one instance of that data structure. The SQL language is used to interact with that database information.

The SQL aggregate functions — AVG, COUNT, DISTINCT, MAX, MIN, SUM — all return a value computed or derived from one column's values, after discarding any NULL values. The syntax of all these functions is:

- **AVG()** The AVG() function calculates the average value of a set of values. It ignores NULL in the calculation.

```
SELECT column1, column2, ... AVG (column1) FROM table_name
```

- **SUM():** The SUM() function returns the sum of values in a set. The SUM() function ignores NULL. If no matching row found, the SUM() function returns NULL.

  To get the total order value of each product, you can use the SUM() function in conjunction with the GROUP BY clause as follows:

```
SELECT column1, column2, ... SUM (column1) FROM table_name
```

- **MAX():** The MAX() function returns the maximum value in a set. For example, you can use the MAX() function to get the highest buy price from the products table as shown in the following query:

```
SELECT column1, column2, ... MAX (column1) FROM table_name
```

- **MIN():** The MIN() function returns the minimum value in a set of values. Code language: MySQL (Structured Query Language) (MySQL) For example, the following query uses the MIN() function to find the lowest price from the products table:

```
SELECT column1, column2, ... MIN (column1) FROM table_name
```

- **Count():** MySQL count() function returns the total number of values in the expression. This function produces all rows or only some rows of the table based on a specified condition, and its return type is BIGINT. It returns zero if it does not find any matching rows. It can work with both numeric and non-numeric data types.

```
SELECT column1, column2, ... COUNT (column1) FROM table_name
```

2.2 Using Text /String Functions:)

1. CHAR() : It returns a string made up of the ASCII representation of the decimal value list. Strings in numeric format are converted to a decimal value. Null values are ignored.
2. CONCAT() :It returns argument str1concatenated with argument str2

```
SELECT CONCAT (column1, column12) FROM table_name
```

3. LOWER()/LCASE():It returns argument str, with all letters in lowercase.

```
SELECT LOWER (column1, column12) FROM table_name
```

4. SUBSTR(): Check by yourself.
5. UPPER()/UCASE(): Check by yourself.

6. LTRIM(): Check by yourself.
7. RTRIM(): Check by yourself.
8. TRIM(): Check by yourself.
9. INSTR(): Check by yourself.
10. LENGTH(): Check by yourself.
11. LEFT(): Check by yourself.
12. RIGHT(): Check by yourself.
13. MID(): Check by yourself.

3 Procedure (Implementation in MySQL)

1. **Create a table product_order_info**

- **Insert Data:**

```
CREATE TABLE 'product_order_info'(
product_no int(11) NOT NULL AUTO_INCREMENT,
product_name varchar(255) NOT NULL,
product_type enum('electronics', 'stationary' , 'food' , 'beverage' ) DEFAULT NULL,
product_price FLOAT(10,2) NOT NULL,
product_quantity SMALLINT NOT NULL,
order_date datetime NOT NULL, DEFAULT current_timestamp,
PRIMARY KEY(product_no)
);
```

- **Insert Multiple VALUES at a time:**

```
INSERT INTO product_order_info (product_no, product_name, product_type, product_price,,
product_quantity)
VALUES (101,'Laptop' , 'electronics' ,67000,'1'),
(null,'Mobile','electronics' ,23500,'1'),
(null,'Watch' , 'electronics' ,8650,'2'),
(null,'Butter', 'stationary', 50, '5'),
(null,'Coca-cola','beverage' , 35, '2'),
(null,'Seven-Up', 'beverage', 55, '1');
```

- AVG function

```
SELECT AVG(product_price) avg_product_price FROM product_order_info;
```

OR

```
SELECT AVG(product_price) avg_product_price AS avg_product_price FROM product_order_info;
```

- COUNT function returns the number of the rows in a table.

```
SELECT COUNT(product_no) AS total_order FROM product_order_info;
```

- COUNT function returns the number of the rows in a table.

```
SELECT COUNT() product_type, product_name, product_price FROM product_order_info
GROUP BY product_type= 'electronics';
```

- COUNT function returns the number of the rows of specific items.

```
SELECT COUNT(*) product_type, product_name, product_price FROM product_order_info
WHERE product_type= 'electronics';
```

- To get the total sales of each product,

```
SELECT product_no, product_name, product_price, product_quantity, SUM(product_price ×
product_quantity) AS total_per_product FROM product_order_info GROUP BY product_no;
```

- MAX function returns the maximum value in a set of values.

```
SELECT MAX(product_price) max_price FROM product_order_info;
```

- MIN function returns the minimum value in a set of values.

```
SELECT MIN(product_price) max_price FROM product_order_info;
```

2. **Using LENGTH(), UCASE/ UPPER CASE(), LCASE/ LOWER CASE(), MID(), ROUND/ FLOOR/ CELLING(), CONCAT():**

- **MySQL LENGTH function**

```
SELECT product_no,product_name,product_price,
LENGTH(product_price) FROM product_order_info ;
```

Example-2:

```
SELECT product_no,product_name,product_price
FROM product_order_info WHERE LENGTH(product_price)>5;
```

- **UCASE function**

```
SELECT product_no,product_name,product_price,
UCASE(product_price) FROM product_order_info ;
```

- **LCASE function**

```
SELECT product_no,product_name,product_price,
LCASE(product_price) FROM product_order_info ;
```

- **FLOOR function**

```
SELECT product_no,product_name,product_price,
FLOOR(product_price) FROM product_order_info ;
```

- **CELLING function**

```
SELECT product_no,product_name,product_price,
CEIL(product_price) FROM product_order_info ;
```

- **ROUND function**

```
SELECT product_no,product_name,product_price,
ROUND(product_price) FROM product_order_info ;
```

- **MID function**

```
SELECT product_no,product_name,product_price,
MID(product_price,1,3) FROM product_order_info ;
```

- **CONCAT function**

```
SELECT product_no,product_name,product_price,
CONCAT(product_name, ' ', product_type) FROM product_order_info ;
```

3. Sorting data using ORDER BY, GROUP BY **try by yourself**

4 Discussion & Conclusion

In summary, this experiment makes a brief analysis of the Aggregate Function implemented by MySQL 8.0 from the source level.Aggregate Function saves the intermediate values of corresponding calculation results without GROUP BY by defining member variables, saves the keys and aggregated values of corresponding GROUP BY by using Temp Table with GROUP BY, and introduces the optimization methods of some Aggregate Functions.Of course, there are two important types of aggregation here: ROLL UP and WINDOWS functions, which will be introduced separately in future chapters due to space limitations.I hope this article can help readers understand the implementation of MySQL Aggregate Function.

5 Lab Task (Please implement yourself and show the output to the instructor)

- Task-1:

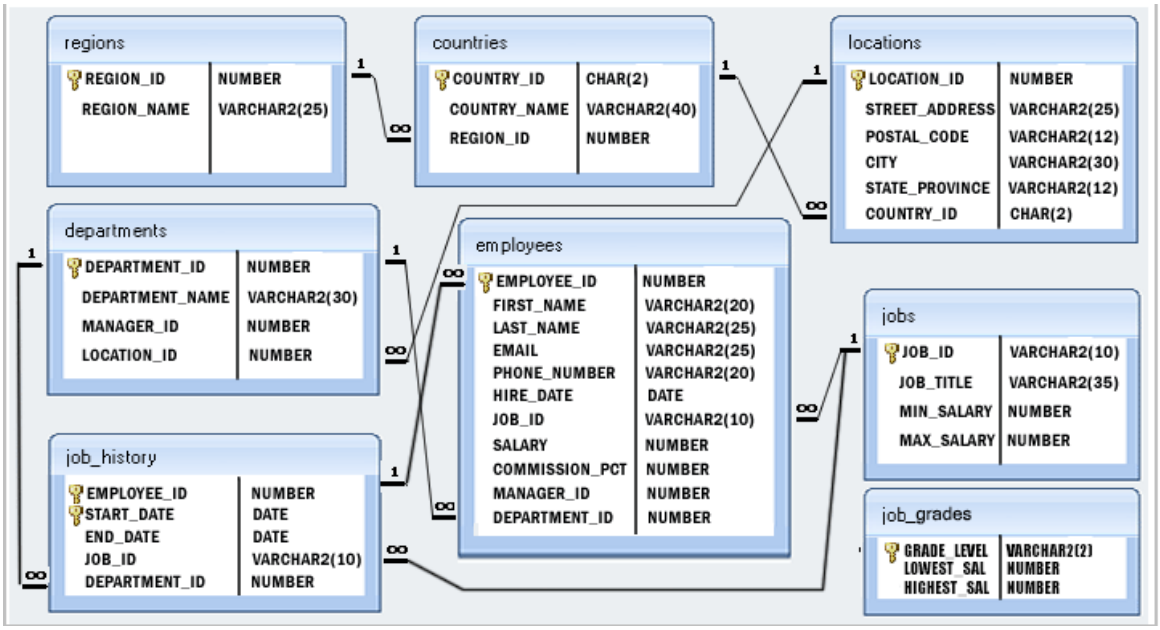

Figure 1: Employees Table Information

1. Input multiple data existing employees database or your existing database table.
2. Write a SQL query for searching employees average age, maximum, minimum salary.
3. Write a SQL statement to find the average purchase amount of all orders.
4. Implement UCASE, LCASE, MID, FLOOR, CELLING, LENGTH function.
5. Which department are paid most and which department are paid less Salary?

6 Lab Exercise (Submit as a report)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24000.00	0.00	0	
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17000.00	0.00	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17000.00	0.00	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	1987-06-20	IT_PROG	9000.00	0.00	102	60
104	Bruce	Ernst	BERNST	590.423.4568	1987-06-21	IT_PROG	6000.00	0.00	103	60
105	David	Austin	DAUSTIN	590.423.4569	1987-06-22	IT_PROG	4800.00	0.00	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	1987-06-23	IT_PROG	4800.00	0.00	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	1987-06-24	IT_PROG	4200.00	0.00	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	1987-06-25	FI_MGR	12000.00	0.00	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	1987-06-26	FI_ACCOUNT	9000.00	0.00	108	100
110	John	Chen	JCHEN	515.124.4269	1987-06-27	FI_ACCOUNT	8200.00	0.00	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	1987-06-28	FI_ACCOUNT	7700.00	0.00	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	1987-06-29	FI_ACCOUNT	7800.00	0.00	108	100
113	Luis	Popp	LPOPP	515.124.4567	1987-06-30	FI_ACCOUNT	6900.00	0.00	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	1987-07-01	PU_MAN	11000.00	0.00	100	30
115	Alexander	Khoo	AKHOO	515.127.4562	1987-07-02	PU_CLERK	3100.00	0.00	114	30
116	Shelli	Baida	SBAIDA	515.127.4563	1987-07-03	PU_CLERK	2900.00	0.00	114	30
117	Sigal	Tobias	STOBIAS	515.127.4564	1987-07-04	PU_CLERK	2800.00	0.00	114	30
118	Guy	Himuro	GHIMURO	515.127.4565	1987-07-05	PU_CLERK	2600.00	0.00	114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	1987-07-06	PU_CLERK	2500.00	0.00	114	30
120	Matthew	Weiss	MWEISS	650.123.1234	1987-07-07	ST_MAN	8000.00	0.00	100	50
121	Adam	Fripp	AFRIPP	650.123.2234	1987-07-08	ST_MAN	8200.00	0.00	100	50
122	Payam	Kaufling	PKAUFLIN	650.123.3234	1987-07-09	ST_MAN	7900.00	0.00	100	50
123	Shanta	Vollman	SVOLLMAN	650.123.4234	1987-07-10	ST_MAN	6500.00	0.00	100	50
124	Kevin	Mourgos	KMOURGOS	650.123.5234	1987-07-11	ST_MAN	5800.00	0.00	100	50
125	Julia	Nayer	JNAYER	650.124.1214	1987-07-12	ST_CLERK	3200.00	0.00	120	50
126	Irene	Mikkilineni	IMIKKILI	650.124.1224	1987-07-13	ST_CLERK	2700.00	0.00	120	50
127	James	Landry	JLANDRY	650.124.1334	1987-07-14	ST_CLERK	2400.00	0.00	120	50
128	Steven	Markle	SMARKLE	650.124.1434	1987-07-15	ST_CLERK	2200.00	0.00	120	50
129	Laura	Bissot	LBISSOT	650.124.5234	1987-07-16	ST_CLERK	3300.00	0.00	121	50
130	Mozhe	Atkinson	MATKINSO	650.124.6234	1987-07-17	ST_CLERK	2800.00	0.00	121	50
131	James	Marlow	JAMRLOW	650.124.7234	1987-07-18	ST_CLERK	2500.00	0.00	121	50
132	TJ	Olson	TJOLSON	650.124.8234	1987-07-19	ST_CLERK	2100.00	0.00	121	50
133	Jason	Mallin	JMALLIN	650.127.1934	1987-07-20	ST_CLERK	3300.00	0.00	122	50

Figure 2: Employees Table Information

1. Write a query to list the number of jobs available in the employees table.
2. Write a query to get the minimum salary from employees table.
3. Write a query to get the maximum salary of an employee working as a Programmer.
4. Write a query to get the average salary for each job ID excluding programmer.
5. Attach with query codes and with output screenshots in the report.

7 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.