LAB 03 SQL Functions

OBJECTIVE(S)

- Learn column placement
- Learn string functions
- Learn math functions
- Learn SQL aggregate functions
- Learn about grouping rows

NEW COLUMN PLACEMENT

If we wish to alter the table by adding new columns, we can decide where to place them in the table. The following keywords allow us to decide their placement.

FIRST

The new column is added as the first column of the table.

- **ALTER TABLE** tb_name **ADD** col_name datatype(size) **FIRST**;

AFTER

This allows us to place the column after another existing column in the table.

- **ALTER TABLE** tb_name **ADD** new_col_name datatype(size) **AFTER** col_name;

TASK

- Add a new column ***sno*** at the beginning of the student's table.
- Add a new column ***guardian_name*** before **the *semester***.

STRING FUNCTIONS

MySQL allows us to manipulate string data using several functions some of which are given below.

UPPER

The **UPPER** or **UCASE** function converts all the characters of a string into uppercase.

- SELECT **UPPER**(col_name) **FROM** tb_name;
- SELECT **UCASE**(col_name) **FROM** tb_name;

LOWER

The **LOWER** or **LCASE** function converts all the characters of a string into lowercase.

- SELECT **LOWER**(col_name) **FROM** tb_name;
- SELECT **LCASE**(col_name) **FROM** tb_name;

LENGTH

The **LENGTH** function returns the length of the data of a particular column.

- **SELECT LENGTH**(col_name) **FROM** tb_name;

CONCAT

The **CONCAT** function adds (concatenates) two or more strings. It returns NULL if any field contains NULL.

- **SELECT CONCAT**(col1_name, col2_name) **FROM** tb_name;
- **SELECT CONCAT**(col1_name, "sample string") **FROM** tb_name;

REPLACE

The **REPLACE** function replaces all instances of a substring within a string.

- **SELECT REPLACE**(col_name, "substring", "replacement") **FROM** tb_name;

TASK

- Display the names of students in upper case.
- Display the names of students in the following format: **SMITH, J.**
- Display the names and departments of students in the following format: **John Smith, CS**.
- Display the names and departments of all the students. Replace all instances of **CS** with **CSE**. Label appropriately.

MATH FUNCTIONS

MySQL also allows us to perform certain mathematical operations on the data of a table. Some functions for these operations are given below. The general syntax of these functions (unless otherwise specified) is:

- **SELECT FUNCTION_NAME**(col_name) **FROM** tb_name;

FUNCTION NAME	DESCRIPTION
ABS	Returns the absolute value of a number.
CEIL	Synonym: **CEILING** Returns the smallest integer value which is greater than the number provided as the argument.
FLOOR	Returns the smallest integer which is lower than the number provided as the argument.
DEGREES	Converts radians into degrees.
RADIANS	Converts degrees into radians.
EXP	Returns the value of e^x, where x is the argument. E.g. **EXP**(x)
SQRT	Returns the square root of the argument.
TRIGONOMETRIC FUNCTIONS	These allow us to perform trigonometric operations on the data of a table. Some common trigonometric functions are **COS()**, **SIN()**, **TAN()**, **ACOS()**, **ASIN()**, and **ATAN()**.

TASK

- Find the absolute value of **-273.15**.
- Convert $\boldsymbol{\pi}$ into degrees.
- Convert **120º** into radians.

DATE/TIME FUNCTIONS

MySQL provides several functions for date/time calculations some of which are listed below. The general syntax of these functions (unless otherwise specified) is:

- **SELECT FUNCTION_NAME**(col_name) **FROM** tb_name;

FUNCTION NAME	DESCRIPTION
YEAR	Extracts the year (YYYY) from a date datatype field.
MONTH	Extracts the month (MM) from a date datatype field.
DAYOFMONTH	Extracts the day (DD) from a date datatype field.
CURDATE	Gives the current date in YYYY-MM-DD format.
TIMESTAMPDIFF	**TIMESTAMPDIFF**(unit, date1, date2) can be used to calculate the time elapsed between date1 and date2 in the specified units. Unit is used to indicate how we want the result to be expressed. It can take the value YEAR, MONTH, or DAY each of which will return the result in the appropriate unit.

TASK

- Display the number of months until your next birthday. Label appropriately.
- Display the year of birth and age of each student. Label appropriately.

AGGREGATE FUNCTIONS

An aggregate function performs a calculation on a set of values and returns a single value. Some common aggregate functions are given below. The general syntax of these functions (unless otherwise specified) is:

- **SELECT AGG_FUNCTION_NAME**(col_name) **FROM** tb_name;

FUNCTION NAME	DESCRIPTION
AVG	Calculates the average for a set of values. **AVG** function ignores NULL values in the calculation.
SUM	Calculates the sum for a set of values. **SUM** function ignores NULL values in the calculation.
MAX	Returns the maximum value from a set of values.
MIN	Returns the minimum value from a set of values.
COUNT	Counts the number of rows in a table. **COUNT**(*) – Counts all rows. **COUNT**(col_name) – Counts non-empty rows. **COUNT**(DISTINCT col_name) – Counts unique rows.

TASK

- Display the average marks of students.
- Display the maximum percentage.
- Display the total number of students who have not yet been assigned a department.

GROUPING ROWS

GROUP BY clause: Groups a set of rows
HAVING clause: Filter condition with GROUP BY clause

GROUP BY

The **GROUP BY** clause is used to group a set of rows of the table. The **GROUP BY** clause is often used with an aggregate function to perform calculations and returns a single value for each subgroup.

- **SELECT** col_name(s), aggregateFunc() **FROM** tb_name
  GROUP BY col1_name, col2_name;

HAVING

The **HAVING** clause is often used with the **GROUP BY** clause. We can apply a filter condition to the columns that appear in the **GROUP BY** clause.

- **SELECT** col_name(s), aggregateFunc() **FROM** tb_name
  GROUP BY col1_name, col2_name
  HAVING group condition(s);

Illegal Queries Using Group Functions

- Group functions CAN NOT be in the **WHERE** clause.
- **WHERE** clause CAN NOT be used to restrict groups.
- **HAVING** clause is used to restrict groups.

TASK

- Display the number of students in each department.
- Display the maximum percentage for each semester. List in order of decreasing semesters.
- List the courses with more than two active enrollments.

LAB ASSIGNMENT

Use the **Employees** table created in Lab 02 assignment for this assignment.

1. Delete the columns ***first_name*** and ***last_name***. Show the table structure.
2. Add a new column ***employee_name*** as the second field. Show the table structure.
3. Display the name of employees along with their designation and departments. The output should have only **one** column with comma-separated values.
4. Count the total number of Auditors in the organization. Label appropriately.
5. List the departments in each city along with their number of employees.
6. List the average salary for each department.
7. Display the number of employees who have served for over 8 years in each department.
8. Calculate the total salary expense for each designation. List from highest to lowest.
9. Count the total number of records. Label the result as **Number of Employees**.
10. For each city, list the departments with an average salary of 70,000 or more. The average salary for these departments should have a precision of two decimal places. Label the columns appropriately.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command-line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **'cs181xxx_Lab03'**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.