/\*=====================================

Section 11: Functions

=======================================\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Aggregate Functions

2. Comparison Functions

3. Date Functions

4. String Functions

5. Window Functions

6. Math Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- ============= 1. Aggregate Functions ================

-- AVG() - Return the average of non-NULL values.

SELECT \* FROM products;

SELECT AVG(buyPrice) as average\_buy\_price

FROM products;

SELECT productLine, AVG(buyPrice) as average\_buy\_price

FROM products

GROUP BY productLine;

-- COUNT()- returns the number of the value in a set.

SELECT COUNT(\*) AS total

FROM products;

SELECT productLine,COUNT(\*) AS total

FROM products

GROUP BY productLine;

-- SUM() - returns the sum of values in a set.

SELECT SUM(amount) as TotalPayment

FROM payments;

SELECT YEAR(paymentDate) as PaymentYear, SUM(amount) as TotalPayment

FROM payments

GROUP BY PaymentYear;

-- MAX() - returns the maximum value in a set.

SELECT MAX(buyPrice) as highest\_price

FROM products;

SELECT productLine, MAX(buyPrice) as highest\_price

FROM products

GROUP BY productLine

ORDER BY highest\_price DESC;

-- MIN() - returns the minimum value in a set.

SELECT MIN(buyPrice) as min\_price

FROM products;

SELECT productLine, MIN(buyPrice) as min\_price

FROM products

GROUP BY productLine

ORDER BY min\_price DESC;

-- GROUP\_CONCAT() - concatenates a set of strings and returns the concatenated string.

SELECT \* FROM customers;

SELECT country, GROUP\_CONCAT(DISTINCT customername ORDER BY customerName) as customer\_list

FROM customers

GROUP BY country;

-- ============= 2. Comparison Functions ================

-- COALESCE() - allows you to substitute NULL values.

SELECT \* FROM customers;

SELECT customerName, city, COALESCE(state, 'N/A') as State, country

FROM customers;

-- GREATEST and LEAST - functions to find the greatest and smallest values of two or more columns respectively.

SELECT MAX(buyPrice), MIN(buyPrice) FROM products;

SELECT ProductName, buyPrice, MSRP FROM products;

SELECT productName,buyPrice, MSRP, GREATEST(buyPrice, MSRP), LEAST(buyPrice, MSRP) FROM products;

-- ISNULL() -- takes one argument and tests whether that argument is NULL or not.

-- The ISNULL function returns 1 if the argument is NULL, otherwise, it returns 0.

SELECT customerName, city, state,ISNULL(state) as `IsStateNULL`, country

FROM customers;

-- ============= 3. Date Functions ================

-- CURDATE() - returns the current date as a value in the 'YYYY-MM-DD' format

SELECT CURDATE();

SELECT CURRENT\_DATE(),CURRENT\_DATE, CURDATE();

-- NOW() - returns the current date and time in the configured time zone as a string or a number

-- in the 'YYYY-MM-DD HH:MM:DD' format.

SELECT NOW();

-- To get in numeric form

SELECT NOW() + 0;

-- SYSDATE()

SELECT SYSDATE();

-- NOW() vs SYSDATE()

SELECT NOW(), SLEEP(5), NOW(); -- constant date and time at which the statement started executing

SELECT SYSDATE(), SLEEP(5), SYSDATE(); -- changes. exact time at which the statement executes

-- DAY() - to get the day of the month of a specified date.

-- MONTH() - returns an integer that represents the month of a specified date value.

-- YEAR() - to get the year out of a date value.

SELECT

\*,

CURRENT\_DATE(),

DAY(paymentDate),

MONTH(paymentDate),

YEAR(paymentDate)

FROM payments;

-- DATEDIFF - calculates the number of days between two DATE, DATETIME, or TIMESTAMP values.

SELECT \* FROM payments;

SELECT

\*,CURRENT\_DATE(),

DATEDIFF(CURRENT\_DATE,paymentDate) as Days,

DATEDIFF(CURRENT\_DATE,paymentDate)/365 as Years

FROM payments;

-- DATE\_ADD - to add a time value to a DATE or DATETIME value.

-- DATE\_ADD(start\_date, INTERVAL expr unit);

SELECT

\*,

CURRENT\_DATE(),

DAY(paymentDate),

MONTH(paymentDate),

YEAR(paymentDate),

DATEDIFF(CURRENT\_DATE,paymentDate) as Days,

DATEDIFF(CURRENT\_DATE,paymentDate)/365 as Years,

DATE\_ADD(paymentdate, INTERVAL 1 DAY)

FROM payments;

SELECT \*, DATE\_ADD(paymentdate, INTERVAL 1 DAY) FROM payments;

SELECT \*, DATE\_ADD(paymentdate, INTERVAL 1 HOUR) FROM payments; -- 2004-10-19 00:00:00 --> 2004-10-19 01:00:00

SELECT \*, DATE\_ADD(paymentdate, INTERVAL 1 MINUTE) FROM payments; -- 2004-10-19 00:00:00 --> 2004-10-19 00:01:00

SELECT \*, DATE\_ADD(paymentdate, INTERVAL 1 SECOND) FROM payments; -- 2004-10-19 00:00:00 --> 2004-10-19 00:00:01

SELECT \*, DATE\_ADD(paymentdate, INTERVAL '-1 5' DAY\_HOUR) FROM payments; -- 2004-10-19 --> 2004-10-18 --> 2004-10-17 19:00:00

SELECT \*, DATE\_ADD(paymentdate, INTERVAL '-1 5' HOUR\_MINUTE) FROM payments;

SELECT \*, DATE\_ADD(paymentdate, INTERVAL '-1 5' SECOND\_MICROSECOND) FROM payments;

SELECT \*, DATE\_ADD(paymentdate, INTERVAL 1 WEEK) FROM payments;

SELECT \*, DATE\_ADD(paymentdate, INTERVAL -1 MONTH) FROM payments;

-- DATE\_SUB() - subtracts a time value (or an interval) from a DATE or DATETIME value.

SELECT \*, DATE\_SUB(paymentdate, INTERVAL 1 DAY) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL 1 HOUR) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL 1 MINUTE) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL 1 SECOND) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL '-1 5' DAY\_HOUR) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL '-1 5' HOUR\_MINUTE) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL '-1 5' SECOND\_MICROSECOND) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL 1 WEEK) FROM payments;

SELECT \*, DATE\_SUB(paymentdate, INTERVAL -1 MONTH) FROM payments;

-- DATE\_FORMAT - to format the date.

SELECT \*, DATE\_FORMAT(paymentdate, '%a') FROM payments;

SELECT \*, DATE\_FORMAT(paymentdate, '%e/%c/%Y') FROM payments;

-- DAYNAME - to get the name of a weekday for a given date.

SELECT \*, DAYNAME(paymentdate) FROM payments;

SELECT

DAYNAME(orderdate) as weekday,

COUNT(\*) as total\_orders

FROM orders

WHERE YEAR(orderdate) = 2004

GROUP BY weekday

ORDER BY total\_orders DESC;

-- DAYOFWEEK - returns the weekday index for a date i.e., 1 for Sunday, 2 for Monday, … 7 for Saturday.

SELECT \*,

DAYNAME(paymentdate),

DAYOFWEEK(paymentdate)

FROM payments;

-- EXTRACT() - extracts part of a date.

SELECT \*, EXTRACT(WEEK from paymentdate) FROM payments;

SELECT \*, EXTRACT(MONTH from paymentdate) FROM payments;

SELECT \*, EXTRACT(DAY from paymentdate) FROM payments;

SELECT \*, EXTRACT(QUARTER from paymentdate) FROM payments;

SELECT \*, EXTRACT(YEAR\_MONTH from paymentdate) FROM payments;

SELECT EXTRACT(YEAR FROM CURDATE());

-- LAST\_DAY() - takes a DATE or DATETIME value and returns the last day of the month for the input date.

SELECT \*, LAST\_DAY(paymentdate) FROM payments;

-- STR\_TO\_DATE() - converts the str string into a date value based on the fmt format string.

SELECT STR\_TO\_DATE('22,2,2022','%d,%m,%Y');

SELECT STR\_TO\_DATE('1,1,2022 is the New Year date','%d,%m,%Y');

SELECT STR\_TO\_DATE('20130101 1130','%Y%m%d %h%i') ; -- refer to DATE\_FORMAT function for the list of format specifiers.

-- TIMEDIFF & TIMESTAMPDIFF - returns the difference between two TIME or DATETIME values.

SELECT TIMEDIFF('12:00:00','10:00:00') as diff;

SELECT TIMEDIFF((NOW() - INTERVAL 1 HOUR), NOW());

SELECT TIMESTAMPDIFF(MONTH, '2012-03-01', NOW());

SELECT TIMESTAMPDIFF(WEEK, '2012-03-01', NOW());

SELECT TIMESTAMPDIFF(DAY, '2012-03-01', NOW());

SELECT TIMESTAMPDIFF(HOUR, '2012-03-01', NOW());

SELECT TIMESTAMPDIFF(MINUTE, '2012-03-01', NOW());

SELECT TIMESTAMPDIFF(SECOND, '2012-03-01', NOW());

-- WEEK - to get the week number for a date.

-- WEEKDAY -- 0 for Monday, 1 for Tuesday, … 6 for Sunday.

-- A year has 365 days for a normal year and 366 days for leap year.

-- A year is then divided into weeks with each week has exact 7 days.

-- So for a year we often has 365 / 7 = 52 weeks that range from 1 to 52.

SELECT \*, WEEK(paymentdate), WEEKDAY(paymentdate) FROM payments;

-- ============= 4. String Functions ================

-- CONCAT & CONCAT\_WS

SELECT CONCAT(contactFirstName,' ',contactLastName) Fullname

FROM customers;

SELECT CONCAT\_WS('/',contactFirstName,contactLastName, City, Country) Fullname

FROM customers;

-- String Length

SELECT productName, LENGTH(productName) FROM products;

-- LEFT & RIGHT

SELECT productName, LEFT(productName, 4), RIGHT(productName, 4) FROM products;

-- INSTR - to return the position of the first occurrence of a string.

SELECT productName, INSTR(productName,'son') FROM products;

-- LOWER & UPPER

SELECT productName, LOWER(productName), UPPER(productName) FROM products;

-- LTRIM, RTRIM, TRIM

SELECT

contactFirstName,

INSTR(contactFirstName,' '),

LENGTH(contactFirstName),

LENGTH(RTRIM(contactFirstName)), -- Trim RIGHT side spaces

LENGTH(LTRIM(contactFirstName)), -- Trim LEFT side spaces

LENGTH(TRIM(contactFirstName)) -- Trim spaces on BOTH sides

FROM Customers WHERE contactFirstName IN ('Carine ','Mary ','Jean');

-- REPLACE

-- REPLACE(str,old\_string,new\_string);

SELECT productName, REPLACE(productName, 'son', ' S-O-N ') FROM products;

-- SUBSTRING

SELECT productName, SUBSTRING(productName,5) FROM products;

SELECT productName, SUBSTRING(productName,10) FROM products;

SELECT productName, SUBSTRING(productName,10,15) FROM products;

SELECT productName, SUBSTRING(productName,-7) FROM products;

SELECT SUBSTRING('Harley Davidson', 5,9); -- starting from the 5th position fetch next 9 characters

-- ============= 5. Window Functions ================

DROP TABLE IF EXISTS sales;

CREATE TABLE sales(

sales\_employee VARCHAR(50) NOT NULL,

fiscal\_year INT NOT NULL,

sale DECIMAL(14,2) NOT NULL,

PRIMARY KEY(sales\_employee,fiscal\_year)

);

INSERT INTO sales(sales\_employee,fiscal\_year,sale)

VALUES('Bob',2016,100),

('Bob',2017,150),

('Bob',2018,200),

('Alice',2016,150),

('Alice',2017,100),

('Alice',2018,200),

('John',2016,200),

('John',2017,150),

('John',2018,250);

SELECT \* FROM sales;

-- ROW\_NUMBER

SELECT

sales\_employee,

fiscal\_year,

sale,

ROW\_NUMBER() OVER() as sales\_rowNum

FROM Sales;

SELECT

sales\_employee,

fiscal\_year,

sale,

ROW\_NUMBER() OVER() as sales\_rowNum,

ROW\_NUMBER() OVER(ORDER BY fiscal\_year) as sales\_rowNum\_orderby,

ROW\_NUMBER() OVER(PARTITION BY fiscal\_year ORDER BY sale DESC) as sales\_rowNum\_part\_orderby

FROM Sales;

-- RANK & DENSE\_RANK

SELECT

sales\_employee,

fiscal\_year,

sale,

ROW\_NUMBER() OVER(PARTITION BY fiscal\_year ORDER BY sale DESC) as sales\_rowNum\_part\_orderby,

RANK() OVER (PARTITION BY fiscal\_year ORDER BY sale DESC) as sales\_rank,

DENSE\_RANK() OVER (PARTITION BY fiscal\_year ORDER BY sale DESC) as sales\_dense\_rank

FROM sales;

-- NTILE - divides rows in a sorted partition into a specific number of groups.

SELECT

sales\_employee,

fiscal\_year,

sale,

NTILE(3) OVER (ORDER BY sale DESC) as sales\_Ntile

FROM sales;

-- LEAD & LAG

SELECT

sales\_employee,

fiscal\_year,

sale,

LEAD(sales\_employee, 1) OVER (ORDER BY sale DESC) as lead\_sales,

LEAD(sales\_employee, 2) OVER (ORDER BY sale DESC) as lead\_sales

FROM sales;

SELECT

sales\_employee,

fiscal\_year,

sale,

LAG(sales\_employee, 1) OVER (ORDER BY sale DESC) as lead\_sales,

LAG(sales\_employee, 2) OVER (ORDER BY sale DESC) as lead\_sales

FROM sales;

-- ============= 6. Math Functions ================

SELECT

\*,

amount\*-1,

ABS(amount\*-1),

CEIL(amount),

FLOOR(amount),

ROUND(amount,1),

MOD(amount,3) -- Returns the remainder of a number divided by another

FROM payments;