ANSI SQL

SQL Functions

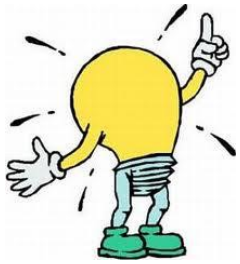
LEVEL – LEARNER

Icons Used

Hands-on Exercise

Reference

Questions



Points To Ponder

Coding Standards

Lend A Hand

Summary

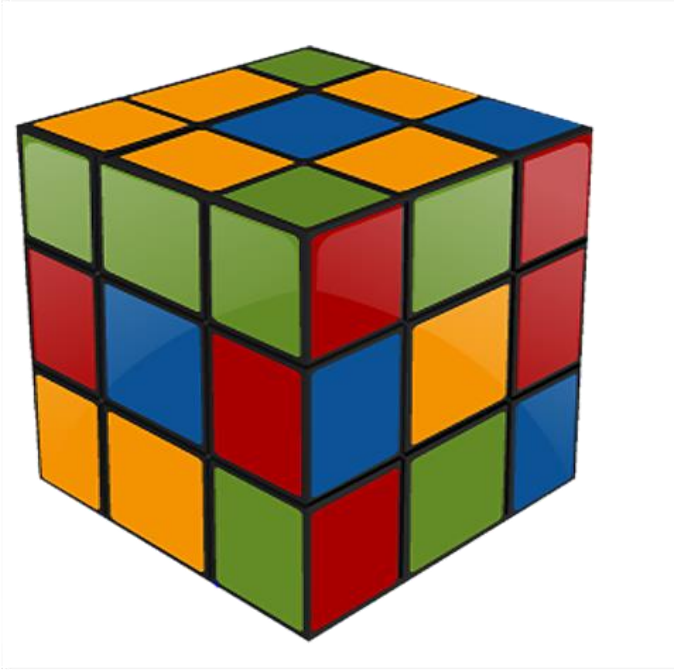
Test Your Understanding

Overview

This session on SQL Functions provides knowledge and understanding of the use of functions available in ANSI. It also demonstrates the application of the syntax which will be taught, as part of this session, in a case study provided.

Objectives

- After completing this session, you will be able to:
  - List the types of SQL Function.
  - Identify the Numeric, Character, and Date Time functions.
  - Describe the Aggregate Function.
  - Describe the Mathematical Function.
  - Describe the Nesting of Functions.



Scenario

- To understand ANSI SQL in detail, we are going to make use of **Product Management System** (**PMS**) for ABC Traders.
- ABC Traders is a company, which buys collectible model cars, trains, trucks, buses, trains and ships directly from manufacturers and sells them to distributors across the globe. This software has been developed in order to manage the stocking, supply and payment transactions.
- As per the requirement of the trading company, an inventory system has been developed to collect the information of products, customers, and their payment processing.

Database Tables

- There are many entities involved in Product Management System, as given below. We will be dealing with these throughout this course.

Offices
To maintain information of offices . For example, office code, address, city, and so on.

Customer
To maintain customer details . For example, customer name, address.

Employees
To maintain employee details. For example, ID, name, and so on.

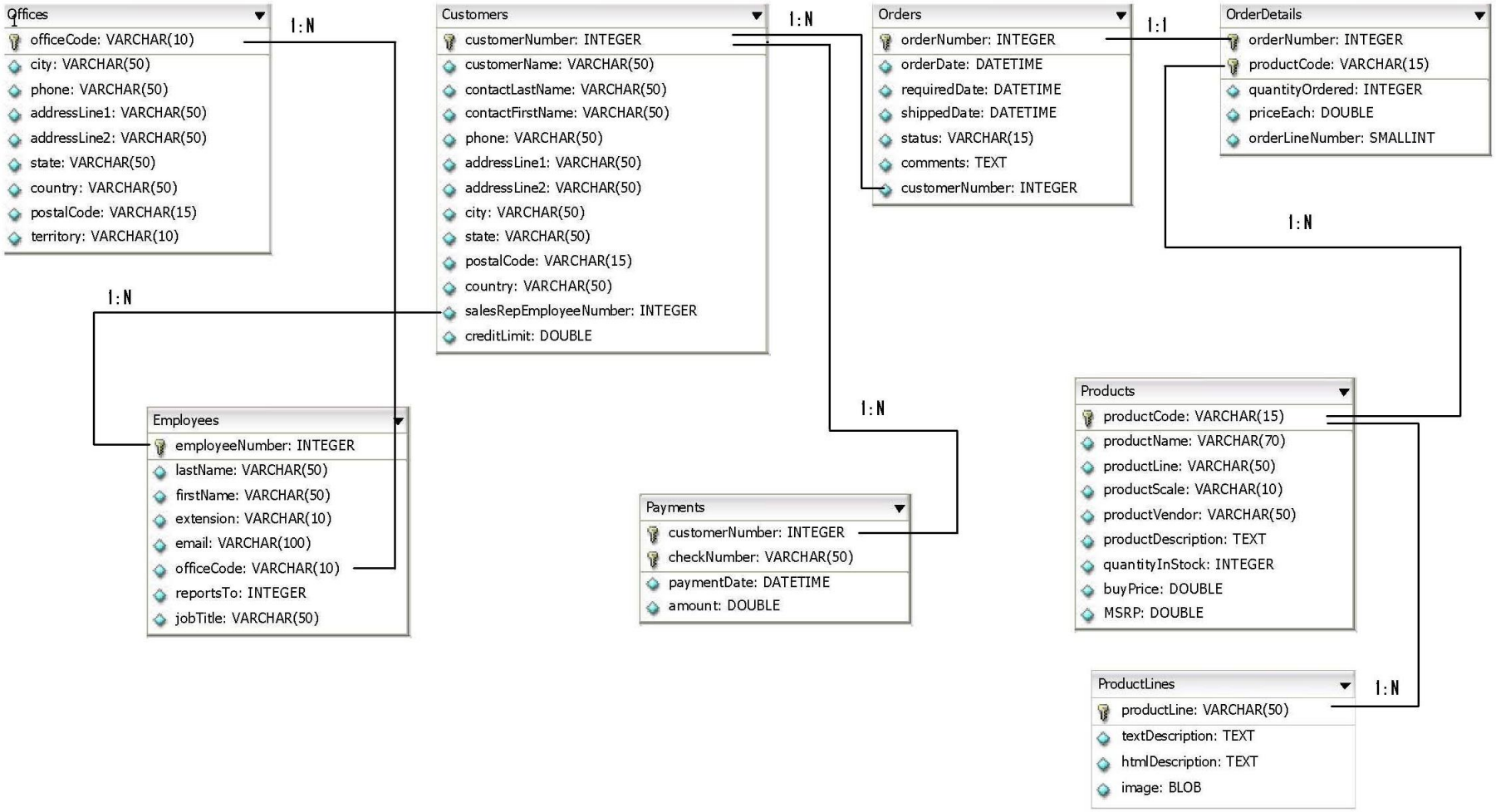
Products
To maintain information of products. For example, product ID, name , and so on.

Payments
To maintain information of the payments done. For example, payment date, amount, and so on.

Orders
To maintain orders placed by customers . For example, order no, date, and so on.

Order Details
To maintain order details done by customers. For example, order no, date, products ordered, price, and so on.

Schema Diagram



Scenario

Hi!
It's good that the requirements like adding two columns are implemented using operators. The remaining requirements like finding the maximum amount paid by the customers should be implemented as well.
But how?

Let's learn about aggregate functions which will help us meet Tim's requirements.

What are Functions?

- What are Functions?

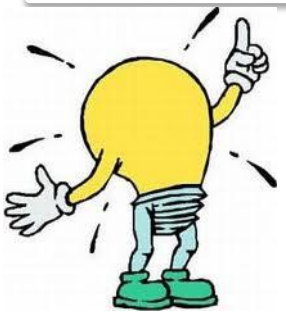
- SQL functions are built in APIs, which SQL provides for developers, and can be used in SQL statements to perform specific logic or functionality.

- **Example:**
  - Round the numbers.
  - Change the string to upper case.

Classifying SQL Functions

- The following points cover the ANSI SQL Functions Classification:
- The first level of classification hierarchy includes:
  - Deterministic functions
  - Non-Deterministic functions
- There are no ironclad rules for recognizing a SQL routine as either deterministic or non-deterministic.
- A Deterministic function always returns the same results if given the same input values.
- A Non deterministic function returns different results every time it is called, even when the same input values are provided.
- An example of a deterministic function is the function LENGTH. When argument of a string data type is passed, it returns the length of the argument passed. Calling it with the same argument over and over again will yield exactly the same result.

A function that takes no arguments is called **Niladic.**



Aggregate Functions and Scalar Functions

- Let us continue with ANSI SQL Functions Classification.
- The SQL has two basic types of functions:
  - Aggregate Functions
  - Scalar Functions
- Aggregate functions operate on sets of rows, and returns one value per group.

aggregate_function_name ([ALL | DISTINCT] expression)

Function	Usage
AVG(expression)	Computes the average value of a column by the expression
COUNT(expression)	Counts the rows defined by the expression
COUNT(*)	Counts all rows in the specified table or view
MIN(expression)	Finds the minimum value in a column by the expression
MAX(expression)	Finds the maximum value in a column by the expression
SUM(expression)	Computes the sum of column values by the expression

Aggregate Function Examples

- Aggregate functions are commonly used with the GROUP BY clause in a SELECT statement, and accept single column as the input.
- The following are some of the Aggregate Functions.

Function	Example	Description
COUNT	SELECT COUNT(CustomerNumber) FROM Customers ;	Displays the total number of rows in the Customers table.
SUM	SELECT SUM(amount) FROM Payments;	Displays the sum of all the payments in the Payments table.
MIN	SELECT MIN(amount) FROM PAYMENTS;	Displays the minimum amount paid in the Payments table.
MAX	SELECT MAX(amount) FROM PAYMENTS ;	Displays the maximum amount paid in the Payments table.
AVG	SELECT AVG(amount) FROM PAYMENTS;	Displays the average of all the amounts paid in Payments table.

Classifying Aggregate Function

- The sub-classing of the aggregate functions is given below:

Category	Usage
Group	These functions deal with the grouping operation or the group aggregate function.
Window	These functions compute their aggregate values like a group function, except that they aggregate over the window frame of a row and not over a group of grouped table.
Unary Group	These functions take an arbitrary <value expression> as an argument.
Binary Group	These functions take a pair of arguments, a dependent one and an independent one, both of which are numeric expressions. They remove NULL values from the group and if there are no remaining rows, they evaluate to 0.
Inverse Distribution	There are only two inverse functions: PERCENTILE_CONT and PERCENTILE_DISC. Both functions take an argument between 0 and 1.
Hypothetical Set	These functions are related to the window functions RANK, DENSE_RANK, PERCENT_RANK and CUME_DIST.

Scenario

Hi!
Write a query to display the customer and employee names in capital letters.

Let us learn about string functions which will help us meet Tim's requirements.

Scalar Functions

- Here are the characteristics of Scalar Functions:
  - Scalar functions either require no arguments, or at most one argument, to be passed to them. They return a single value that is based on the input value.
  - Scalar functions can be broken down into the subcategories shown in the following table, based upon their intended use.

Category	Usage
Built-in	These functions perform operations on values and settings that are built into the database (such as specifics dealing with the user session)
String	These functions perform operations on character values such as CHAR and VARCHAR and they can return either numeric or string values.
Numeric/Mathematic	These functions perform operations on numeric values.
Date and Time	These functions perform operations on date/time fields.
CASE and CAST	CASE supplies IF-THEN logic to SQL statements and CAST can convert values from one data type to another.

Built-in Scalar Functions

- Here are the characteristics of Built-in Scalar functions:
  - Built-in scalar functions identify both the current user session and the characteristics of the current user session, such as the current session privileges.
  - Built-in scalar functions are always non deterministic.

Function	Usage
CURRENT_DATE	Identifies the current date.
CURRENT_TIME	Identifies the current time.
CURRENT_TIMESTAMP	Identifies the current date and time.
CURRENT_USER	Identifies the currently active user within the database server.
SESSION_USER	Identifies the currently active Authorization ID, if it differs from the user.
SYSTEM_USER	Identifies the currently active user within the host operating system.

String Functions

- Here are the characteristics of String Functions:
  - String function accepts character value as input and can return both character and numeric value.
- Here are a few examples of String Functions:

Function	Usage
CONCATENATE	Appends two or more literal expressions, column values, or variables together into one string.
CONVERT	Converts a string to a different representation within the same character set.
LOWER	Converts a string to all lowercase characters.
SUBSTRING	Extracts a portion of a string.
TRANSLATE	Converts a string from one character set to another.
TRIM	Removes leading characters, trailing characters, or both from a character string.
UPPER	Converts a string to all uppercase characters.

String Function Examples

- Here are some examples of some more String Functions:

Function	Description	Example	Result
UPPER	Converts Alpha Character values to Upper Case	SELECT UPPER (CUSTOMENAME) FROM CUSTOMERS;	ATELIER GRAPHIQUE
LOWER	Converts Alpha Character values to Lower Case	SELECT LOWER(CUSTOMENAME) FROM CUSTOMERS;	atelier graphique
CONCAT	Concatenates the first character value to the second character value.	SELECT CONCAT (CUSTOMERNUMBER, CUSTOMENAME) FROM CUSTOMERS;	103Atelier graphique
SUBSTRING	Returns the specified characters from the character starting position and retrieve the next n characters.	SELECT SUBSTRING(CUSTOMENAME1,3) FROM CUSTOMERS;	Ate
TRIM	Enable you to trim leading or trailing (or both) from a character string	SELECT TRIM(LEADING 'A' FROM CUSTOMENAME) FROM CUSTOMERS SELECT TRIM(TRAILING 'e' FROM CUSTOMENAME) FROM CUSTOMERS;	telier graphique telier graphiqu

Scenario

Hi!
Now that you have converted the names to upper case, round the number to two decimal points.

Let us learn about numeric functions which will help us meet Tim's requirements.

Numeric/Mathematical Functions

- Numeric/Mathematical Functions:
  - Mathematic functions accept numbers, process it and return numeric values.

Function	Usage
ABS	Returns the and absolute value of n.
CEIL	Returns the smallest integer greater than or equal to n.
FLOOR	Returns the largest integer equal to or less than n.
MOD	Returns the operator remainder of m divided by n; returns m if n is 0.
PI	Returns pi (approx. 3.1415926...).
POWER	Returns m raised to the nth power.
ROUND	Returns a numeric expression rounded to the specified length or precision.
SQRT	Returns the square root of n.
SQUARE	Returns the expression raised to the power of 2; equivalent to POWER (number,2).
TRUNC or TRUNCATE	Returns the number x, truncated to D decimals. If D is 0, the result will have no decimal point or fractional part. If D is negative, the integer part of the number is zeroed out.

Numeric/Mathematical Functions

- A few examples of mathematic functions are shown below:

Function Name	Description	Example	Output
Round	Rounds value to specified decimal.	SELECT CUSTOMERNAME, ROUND (CREDITLIMIT, 2) FROM CUSTOMERS //When Creditlimit=4500.926	4500.93
Truncate	Truncates value to specified decimal	SELECT CUSTOMERNAME, TRUNCATE(CREDITLIMIT, 2) FROM CUSTOMERS //When Creditlimit=4500.926	4500.92
Mod	Returns remainder of division	SELECT CUSTOMERNAME, MOD(CREDITLIMIT, 300) FROM CUSTOMERS //When CreditLimit=1600	100

Scenario

Hi!
I would like to get Today's Date. What is the command to find the current date?

Let us learn about date time functions which will help us meet Tim's requirements.

Date Time Function

- Definition: Date/Time functions operate on date, and timestamp data type.

Function Name	Description	Example	Result
DATE	Converts TIMESTAMP or character string to DATE.	select date(200802) from customers;	2020-08-02
ADDDATE	Adds interval to date time value.	SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY) FROM customers;	2008-02-02
DATEDIFF	Subtract two dates (Computes difference between two date time values.)	SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30') from customers;	1
TIME	Converts TIMESTAMP or character string to TIME.	SELECT time('2008-02-03') FROM customers;	00:20:08
EXTRACT	Allows the date part to be extracted (YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, or TIMEZONE_MINUTE) from a temporal expression.	SELECT EXTRACT(DAY FROM DATE('2009-01-01')) FROM customers;	1

Date Time Function (Contd.)

- Here are a few more examples:

Function Name	Description	Example	Result
CURRENT_DATE	Returns current date	SELECT CURRENT_DATE;	2013-02-15
CURRENT_TIME	Returns current time	SELECT CURRENT_TIME;	09:40:51
CURRENT_TIMESTAMP	Returns current date and time	SELECT CURRENT_TIMESTAMP;	2013-02-15 09:42:40
Date Addition	Adding days to a date	SELECT CURRENT_DATE+10;	20130225
Date Subtraction	Subtracting days from a date	SELECT CURRENT_DATE-10;	20130205
Date Difference	Provides no of days between two dates	SELECT CURRENT_DATE- orderdate FROM orders; (Current date is 2013-02-15 and orderdate is 2013-02-05);	10

Miscellaneous Functions

- Miscellaneous Functions:

Function	Usage
COALESCE	Returns the first non-NULL value of a list, or NULL if there are no non-NULL values.. The list includes column Names/Values.
NULLIF	Compares two expressions; if they are equal, returns NULL; otherwise returns the first expression.

- COALESCE():

```
SELECT COALESCE( column1,column2)

FROM <TABLE-NAME>;
```

- Syntax :
  - The COALESCE() function returns the first non-null expression in the expression list. At least one expression must not be the literal NULL. If all expressions evaluate to NULL, then the function returns NULL.
  - Consider the following example:

```
SELECT coalesce(State, 'Not assigned')
FROM Customers ;
```

 |

NULLIF Function

- The NULLIF function compares two columns.
  If both the columns are equal, the NULLIF function returns NULL. Otherwise, it returns the value of the first column.
- Syntax :
  - Consider the following example:

```
SELECT NULLIF( column1,column2)

FROM <TABLE-NAME>;
```

Column1 and Column 2 must be of the same data type.

Examples:

```
SELECT NULLIF(12, 12)
FROM Customers;
```
would return NULL

```
SELECT NULLIF(12, 13)
FROM Customers;
```
would return 12

```
SELECT NULLIF('apples', 'apples')
FROM Customers;
```
would return NULL

```
SELECT NULLIF('apples', 'oranges')
FROM Customers;
```
would return 'apples'

Control Flow Functions

- Control Flow Functions:
  It is similar to the IF-THEN-ELSE logic where a value is substituted based on the return value of the column.
- Syntax :
  - Consider the following example:

```
CASE value WHEN [compare_value] THEN result [WHEN
[compare_value] THEN  result ...] [ELSE result] END
```

Examples:

```
Select CustomerName, Country,
CASE Country
WHEN 'USA' THEN 'United State
   of America'
WHEN 'UK' THEN 'United
   Kingdom'
ELSE 'N/A' END
FROM Customers
```


CASE Operator Examples

Example 1:

```
SELECT CASE 1 WHEN 1 THEN 'this is case one'
WHEN 2 THEN 'this is case two'
ELSE 'this is not in the case'
END as 'how to execute case statement'
```

Explanation

Since CASE is 1, so "this is case one" is returned.

Example 2:

```
SELECT CASE 'A' WHEN 'a' THEN 1
WHEN 'b' THEN 2 END;
```

Explanation

Since CASE is not satisfied by neither of the WHEN, it returns NULL.

Control Flow Functions

- Here are the features of the Control Flow Functions.
  - If expr1 is TRUE (expr1 <> 0 and expr1 <> NULL) then IF() returns expr2; otherwise it returns expr3.

```
IF(expr1, expr2, expr3)
```

  - IF() returns a numeric or string value, depending on the context in which it is used.

```
IFNULL(expr1, expr2, expr3)
```

  - If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.
  - IFNULL() returns a numeric or string value, depending on the context in which it is used.

Examples:

```
Select IF(1>2,2,3); // 3
Select IF(1<2,'Yes','no'); // Yes
Select IF(STRCMP('hi','h1'),'no',
'yes');
-> no
Select IFNULL(1,0); // 1
Select IFNULL(NULL,10); // 10
```


Nesting Of Functions

- Nesting of Function: In case of nesting of functions the inner most functions is evaluated first and the output of that function serves as input to outer function. The process goes till outermost functions return the value. Scalar functions can be nested to any level, though some database vendors have their own restrictions.

- Example:

```
SELECT AVG(IFNULL(CREDITLIMIT, 0))
FROM CUSTOMERS;
```

Step 1:
- IFNULL function is applied:
- If the SAL column in NULL, it is replaced as 0

Step 2:
- AVG function is applied:
- Average is taken after the IFNULL function is applied

SQL Expression

What is Expression?

An expression is a combination of one or more of conditions, values, operators, and SQL functions that evaluate to a value.

- **Where they can be used?**
  - Expressions can be used in:
    - The SELECT statement.
    - A condition of the WHERE,HAVING and ORDER BY clause
    - The VALUES clause of the INSERT statement
    - The SET clause of the UPDATE statement

SQL Expression

- Examples of Expression:

Expression Name	Description	Examples
Simple Expression	A simple expression specifies a column, pseudo column, constant, sequence number, or null.	Buyprice + MSRP
Compound Expression	A compound expression specifies a combination of a function and one or multiple expressions.	creditlimit * AVG(amount)
Date Time Expression	A Date Time Expression can be a date time column or a compound expression that yields a date time value.	(requiredDate – shippeddate)/7

SQL Expression

- Here are some more examples of Expression:

Expression Name	Description	Examples
Function Expression	A Function Expression can be combination of one or more Functions.	`SUM(amount) * AVG(creditlimt) COUNT(customername)`
CASE Expression	It is similar to the IF-THEN-ELSE logic where a value is substituted based on the return value of the column.	`Select customerNumber, country, CASE country WHEN 'USA' THEN 'America' WHEN 'UK' THEN 'Britain' ELSE 'NA' END FROM customers`

Scenario

Now we have implemented few of the Tim's requirements successfully using functions.

Recap of the Case Study

We will use the same CMS case study for learning how to use operators in DQL and DML statements.

- Case Study Scenario:
  - This case study is to develop a *Course Management System* (CMS) for ABC University. The following are the two uses case for which the database needs to be designed.
    - Add Course: To add the course details into the course management system.
    - Retrieve Course: Retrieve the courses stored in the system and display it.
- The courses to be added will have the attributes Course Code, Course Name, Number of participants, Course Description, Course Duration, Course start date and Course Type.

Lend a Hand

Prerequisite : Use the Course_Info and Course_Fees table.

- Insert 2 records in course_fees table with base fees as null.
- Insert 2 records in course_fees table with base fees as 300 and 175.

Problem 1: Write a query which will display the total number of records in Course_Info table.

Problem 2: Develop a query which will give the sum of all base fees of all courses in the Course_Fees table.

Problem 3: Display the minimum and maximum base fees of the courses.

Problem 4: Display the average infra fees of the courses.

Solutions

Solution 1:

```
SELECT COUNT(*)
FROM COURSE_INFO
```

Solution 2:

```
SELECT SUM(BASE_FEES)
FROM course_fees
```

Solution 3:

```
SELECT MIN(BASE_FEES),MAX(BASE_FEES)
FROM COURSE_FEES
```

Solution 4:

```
SELECT AVG(INFRA_FEES)
FROM course_fees
```


Lend a Hand

Pre-requisite: We will use the Course_Info and Course_Fees tables for doing this **Lend a Hand**. Add a new column called Infra_Fees in course_fees with type number(5,3). Update the Infra_Fees of all records with some values say 45.751, 43.453, and so on.

Hints: Use joins wherever needed.

Problem 5 : Develop a query which will display the course name and course Infra fees of all the course. The infra fee should be rounded to one decimal point.

Lend a Hand

Problem 6 : Develop a query which will list all the course codes and course names in the Course_Info table, where the first letter should be a capital letter.

Problem 7: Develop a query which will display the course name and the number of days between the current date and course start date, in Course_Info table

Problem 8: Develop a query which will concatenate the Course Name and Course Code in the following format and display all the courses in the course_info table.

"< Course Name><Course Code>"

Lend a Hand

Problem 9: Develop a query which will display all the Course Names in upper case.

Problem 10: Develop a query which will display all the characters between 1 and 3 of the Course Description column for all the courses in the Course_Info table.

Problem 11: Develop a query to calculate average of all the base fees, consider zero for base fees whose value is Null

Solutions

Solution 5:

```
SELECT COURSE_INFO.COURSE_NAME,
ROUND(COURSE_FEES.INFRA_FEES,2)
FROM COURSE_INFO,COURSE_FEES
WHERE COURSE_INFO.COURSE_CODE=COURSE_FEES.COURSE_CODE
```

Solution 6:

```
SELECT CONCAT(UPPER(LEFT(COURSE_NAME, 1)),
LOWER(SUBSTRING(COURSE_NAME, 2)))
FROM COURSE_INFO
```

Solution 7:

```
SELECT COURSE_NAME,
TO_DAYS(current_date) -
TO_DAYS(course_start_date)
FROM COURSE_INFO
```


Solutions

Solution 8 :

```
SELECT CONCAT(COURSE_NAME,COURSE_CODE)
FROM COURSE_INFO
```

Solution 9 :

```
SELECT UPPER(COURSE_NAME)
FROM COURSE_INFO
```

Solution 10:

```
SELECT SUBSTR(COURSE_DESCRIPTION,1,3)
FROM COURSE_INFO
```

Solution 11:

```
SELECT AVG(IFNULL(BASE_FEES,0))
FROM COURSE_FEES
```


Lend a Hand

Prerequisite : Use the Course_Info and Course_Fees table.

- Insert 2 records in course_fees table with base fees as null.
- Insert 2 records in course_fees table with base fees as 300 and 175.
- Insert 3 records in course_info table each course with course type CLR, EL, OF

Problem 12: Write a query which will display the course code, course type, and the appropriate message as mentioned below.

Course_Type	Message
CLR	'Class Room'
EL	'ELearning'
OF	'Offline Reading'

Solutions

Solution 12:

```
SELECT COURSE_CODE, CASE COURSE_TYPE WHEN 'CLR' THEN
'lCLASS ROOM'

WHEN 'EL' THEN 'ELEARNING'

WHEN 'OF' THEN 'OFFLINE READING'

END RESULT

FROM COURSE_INFO
```


Lend a Hand

Prerequisite: Let us use the ***Student_Info*** and ***Course_Fees*** table.

Problem 13:
Write a query which will convert Student Info's Student_ID to Number, add 100000 and display it for all the students, in the Student_Info table.

Problem 14 :
Write a query which will convert Base_Fees into Varchar from the Course_Fees table.
Display in the following format:
'The Base Fees Amount for the course name' <Course Name>' is '<Base Fees>

Solutions

Solution 13:

```
SELECT 100000 + CAST(STUDENT_ID as decimal)
FROM STUDENT_INFO
```

Solution 14:

```
SELECT CONCAT('THE BASE FEES AMOUNT FOR THE COURSE_NAME IS',
CINFO.COURSE_NAME, CAST(FEESINFO.BASE_FEES as decimal))
FROM COURSE_INFO CINFO,COURSE_FEES FEESINFO
WHERE CINFO.COURSE_CODE=FEESINFO.COURSE_CODE
```


Questions?

Check Your Understanding

What function is used to find the number of characters in Varchar?

What function is used to get the current system date?

What will ROUND (2323.343,2) return?

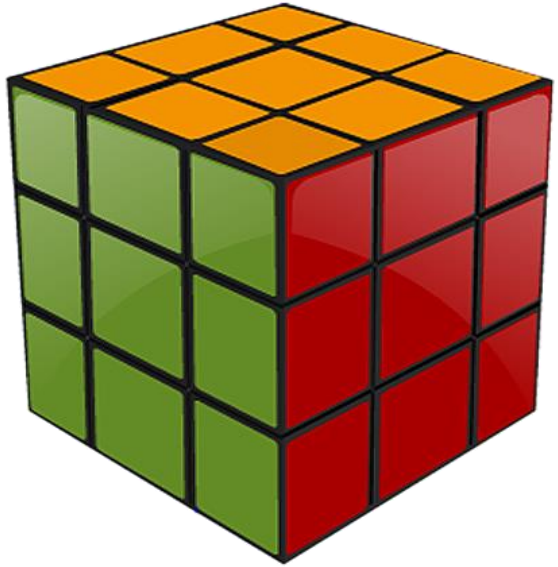
What function is used to get number of years between two dates?

What function is used to convert a character to a number?

What function is used to convert a varchar to a date?

Summary

- The key points covered in this chapter are:
  - ANSI SQL Functions Classification
  - Deterministic and Nondeterministic functions
  - Aggregate Functions and Scalar Functions
  - String Functions, Mathematical Functions
  - Miscellaneous Functions (COALESCE & NULLIF)
  - Nesting of Functions & SQL Expression



Source

Book: O'Reilly SQL In NutShell; **Page No:** 164

Book: Wrox SQL Functions Programmer's Reference 2005; **Page No:** 34

Change Log

Version Number	Changes made			
V1.0	Initial Version			
V1.1	Slide No.	Changed By	Effective Date	Changes Effected
	1-52	Learning Content Team CI Team CATP Technical Team	17-05-2013	Base-lining content

ANSI SQL

You have successfully completed - SQL Functions