Data Types and Functionality

After completing this module, you will be able to:

- **Identify various data types for table columns.**
- **Determine various effects of trailing spaces on character values.**
- **Use CAST to convert from one data type to another.**
- **Use FORMAT to display results in a more desirable form.**
- **Perform various kinds of arithmetic on numeric data.**
- **Use the functions like**
  ABS(*arg*), EXP(*arg*), LOG(*arg*), LN(*arg*), SQRT(*arg*)
- **Use various formatting options on date fields.**
- **Concatenate fields together**

CHARACTER Data Types

There are two basic CHARACTER data types.

- **CHARACTER(n) or CHAR(n)** ***(Where "n" is the number of characters)***
  Fixed Character, Left justified, right padded with spaces to fill the length.
  Example for last name, below is always 20 characters and 20 bytes of storage.

- **VARCHAR(n)** ***(Where "n" is the number of characters)***
  Variable length. Spaces count as valid characters.
  The number of bytes for storage varies according to the value.
  Examples `'this value'` ***has 10 characters and takes 10 bytes of storage.***
  `'this value '` ***has 13 characters and takes 13 bytes of storage.***

BETWEEN Functionality with CHARACTER

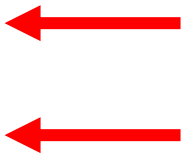
BETWEEN can be used with character data.

Consider what happens when case sensitivity gets involved
(Last_Name is defined as CHAR(20))

WHERE Last_Name BETWEEN 'r' AND 's'

If Not Case Specific: ***(underscore used to represent a space)***

```
'r _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
'R y a n _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
's _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
'S t e i n _ _ _ _ _ _ _ _ _ _ _ _ _ _'
```



BETWEEN this AND this

If Case Specific: ***No rows returned. Why?***

```
'R _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
'R y a n _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
'S _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
'S t e i n _ _ _ _ _ _ _ _ _ _ _ _ _ _'
        .  .  .
'r _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
's _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _'
```

BETWEEN this AND this

Arithmetic Operators

The following arithmetic operations can appear as

- Projected Values
- Conditional Expressions

Teradata Extensions:

Operator	Meaning
**	Exponentiation
MOD	Modulo (remainder)

Operator	Meaning
( )	Evaluated first
*	Multiply
/	Divide
+	Add (positive value)
-	Subtract (negative value)

The order of operations for these are as follows.

- ( )
- Exponentiation
- Multiplication, Division, and Modulo *(left to right)*
- Addition and Subtraction *(left to right)*

Arithmetic and Derived Values

Examples using arithmetic expressions.

In a projection.

SELECT 5 + 2;	→ 7
SELECT 8.40 / 4.20;	→ 2.00
SELECT 10 - 7 * 8;	→ -46
SELECT Salary_Amount * 1.25;	→ Give a 25% raise?
SELECT Salary_Amount - 1e2;	→ Subtract 100 from the budget amount
SELECT 1e6 - 100000;	→ 9.00000000000000E 005

In a conditional Expression.

```
SELECT * FROM Employee WHERE Salary_Amount * 1.25 > 20000;
```

Data Type Conversions Using CAST

You can change the data type for a column or expression by using the CAST function.

The general form of this function is → CAST(*expression* AS *data type*);

Some examples of using CAST follow.

```
SELECT CAST(Last_Name AS CHAR(10)) FROM Employee;
```

```
SEL CAST(3.7777 AS INTEGER);
```
→ this will truncate the decimal to 3

```
SELECT CAST(Budget_Amount * 1.375 AS DEC(15,3)) FROM Department;
```

```
SELECT CAST(1.777 AS CHAR(10));
```
→ this will left-justify 1.777 into a CHAR(10) field

```
SEL CAST(Salary_Amount AS CHAR(11)) FROM Employee;
```

```
SEL CAST(1.77 AS CHAR(2))
```
→ this will truncate the decimal to "1." left justified as a two character field

What should happen to this request?

```
SELECT CAST(Last_Name AS INTEGER) FROM Employee;
```

***** Read the left-hand page to learn how truncation using CAST works in ANSI mode.***

Concatenating Data Types

Concatenation is a method for combining several columns or expressions into a larger, single, field of character data type using an implicit CAST.

Examples of results will be shown on the next page.

Two consecutive pipe characters (or vertical bars) are interpreted by the database as a request to perform a concatenation of fields.

Examples:

```
SELECT Last_Name || First_Name FROM Employee;
SELECT Last_Name || ', ' || First_Name FROM Employee;
SELECT First_Name||' '||Last_Name||' is '||(DATE - Birthdate) / 365 FROM Employee;
SELECT 123||'ABC'||456;
```

In the previous examples, the resulting single field results will all have a data type of VARCHAR.

You can CAST the result of a concatenation like this.

```
SELECT CAST( Last_Name || ', ' || First_Name AS CHAR(100) )
FROM Employee;

SELECT CAST(123||'ABC'||456 AS CHAR(3));  -- This will result in a truncation
```

Concatenated Example Results

```
SELECT Last_Name || ', ' || First_Name FROM Employee WHERE Last_Name = 'Brown';
```

```
((last_name||', ')||first_name)
-------------------------------------------------
Brown                , Allen
Brown                , Alan
```

In the examples, last_name is CHAR(20) and first_name is VARCHAR(30).

```
SELECT First_Name || ', ' || Last_Name FROM Employee WHERE Last_Name = 'Brown';
```

```
((first_name||', ')||last_name)
-------------------------------------------------
Allen, Brown
Alan, Brown
```

```
SELECT 123||'ABC'||456  AS  Concat;
```

```
Concat
----------
123ABC   456
```

In the examples, 123 is data type byteint. *(3 digits plus the sign are a total of 4 characters, right justified)* *What data type is 456?*

```
SELECT CAST(123||'ABC'||456 AS CHAR(3));
```

```
((123||'ABC')||456)
-------------------
 12
```

Module 4: Summary

- Data types can be character, numeric, or just byte string.
- Case sensitivity has an effect on how SQL works with character data.
- Arithmetic can be performed on numeric data.
- Many functions have been created to operate on all data types.
- You can change the data type for any expression by using the CAST function.
- You can use FORMAT to change the formatting a column display.
- Many date formatting options exist to tailor how dates can be displayed.

Module 4: Review Questions

True or False:

1. **The FLOAT data type has more precision than does a decimal data type.**

2. **Character data types can not be converted to a numeric data types.**

3. **FORMAT 'd2' is a valid formatting option.**

4. **The expression → 'a ' = 'A ' evaluates true.**

5. **You can use the CAST function to change a data type or to format results.**

6. **The comma “,” is a valid formatting character.**

7. **The formatting character “9” may be used to display leading or trailing zeroes.**

Module 4: Lab Exercise

1) **Find and list employees first and last names for employees where their last name begins with either "R", "S" or "T".** ***(Do this without regard to case sensitivity. Case sensitivity issues will be discussed in later modules.)***

2) **Write a request that will show the salary amount for the people identified in #1 if they were given a 10% increase in salary that gave them a salary > 50K.**

3) **Use the "Employee Phone" table to project the phone numbers (along with any other columns you desire) for employees having employee numbers between 1005 and 1010. Format the phone number (they do not include an area code) to include a dash in the appropriate place. Make sure that the first part shows 3-digit and that the second part shows 4-digits.**

4) **Project new employee job codes (from the Employee table) for all those job codes ending in 101, increasing them by the size of their department number. Include last names, job codes, department numbers to make help verify results.**

Tricky optional exercise:

5) **Project all salary amounts for employees as if they were rounded to the nearest 100 dollars.** ***(Hint: → Salary_Amount, (((salary_amount) / 100) (DEC(10,0))) will force rounding after the division.)***