EXAMPLES OF ADVANCED FORMULA FIELDS

Summary

Review examples of formula fields for various types of apps that you can use and modify for your own purposes.

Review examples of formula fields for various types of apps that you can use and modify for your own purposes.

This document contains custom formula samples for the following topics. For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Sample Account Management Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Account Rating

This formula evaluates `Annual Revenue`, `Billing Country`, and `Type`, and assigns a value of "Hot," "Warm," or "Cold."

```
IF (AND (AnnualRevenue > 10000000,
CONTAINS (CASE (BillingCountry, "United States", "US", "America", "US",
 "USA", "US", "NA"), "US")),
IF(ISPICKVAL(Type, "Manufacturing Partner"), "Hot",
IF(OR (ISPICKVAL (Type, "Channel Partner/Reseller"),
ISPICKVAL(Type, "Installation Partner")), "Warm", "Cold")),
"Cold")
```

In addition, you can reference this Account Rating formula field from the contact object using cross-object formulas.

```
Account.Account_Rating__c
```

Account Region

This formula returns a text value of "North," "South," "East," "West," or "Central" based on the `Billing State/Province` of the account.

```
IF(ISBLANK(BillingState), "None",
IF(CONTAINS("AK:AZ:CA:HA:NV:NM:OR:UT:WA", BillingState), "West",
IF(CONTAINS("CO:ID:MT:KS:OK:TX:WY", BillingState), "Central",
IF(CONTAINS("CT:ME:MA:NH:NY:PA:RI:VT", BillingState), "East",
IF(CONTAINS("AL:AR:DC:DE:FL:GA:KY:LA:MD:MS:NC:NJ:SC:TN:VA:WV",
BillingState), "South",
IF(CONTAINS("IL:IN:IA:MI:MN:MO:NE:ND:OH:SD:WI", BillingState), "North",
 "Other"))))))
```

Contract Aging

This formula calculates the number of days since a contract with an account was activated. If the contract `Status` is not "Activated," this field is blank.

```
IF(ISPICKVAL(Contract_Status__c, "Activated"),
NOW() - Contract_Activated_Date__c, null)
```

Sample Account Media Service Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

BBC™ News Search

This formula creates a link to a BBC news search site based on the `Account Name`.

```
HYPERLINK(
"http://www.bbc.co.uk/search/news/?q="&Name,
"BBC News")
```

Bloomberg™ News Search

This formula creates a link to an account's ticker symbol on the Bloomberg website.

```
HYPERLINK(
"http://www.bloomberg.com/markets/symbolsearch?query="&TickerSymbol,
"Bloomberg News")
```

CNN™ News Search

This formula creates a link to a CNN news search site using the `Account Name`.

```
HYPERLINK(
"http://http://www.cnn.com/search/?query="&Name,
"CNN News")
```

MarketWatch™ Search

This formula creates a link to an account's ticker symbol on the Marketwatch.com website.

```
HYPERLINK(
"http://www.marketwatch.com/investing/stock/"&TickerSymbol,
"Marketwatch")
```

Google™ Search

This formula creates a link to a Google search site using the `Account Name`.

```
HYPERLINK(
"http://www.google.com/#q="&Name,
"Google")
```

Google News Search

This formula creates a link to a Google news search site using the `Account Name`.

```
HYPERLINK(
"http://news.google.com/news/search?en&q="&Name,
"Google News")
```

Yahoo!™ Search

This formula creates a link to a Yahoo! search site using the `Account Name`.

```
HYPERLINK(
"http://search.yahoo.com/search?p="&Name,
"Yahoo Search")
```

Yahoo! News Search

This formula creates a link to a Yahoo! news search site using the `Account Name`.

```
HYPERLINK(
"http://news.search.yahoo.com/search/news?p="&Name,
"Yahoo News")
```

Sample Case Management Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Autodial

This formula creates a linkable phone number field that automatically dials the phone number when clicked. In this example, replace `"servername"` and `"call"` with the name of your dialing tool and the command it uses to dial. The merge field, `Id`, inserts the identifier for the contact, lead, or account record. The first `Phone` merge field tells the dialing tool what number to call and the last `Phone` merge field uses the value of the `Phone` field as the linkable text the user clicks to dial.

```
HYPERLINK("http://servername/call?id=" & Id & "&phone=" &
                    Phone, Phone)
```

Case Categorization

This formula displays a text value of "RED," "YELLOW," or "GREEN," depending on the value of a case age custom text field.

```
IF(DaysOpen__c > 20, "RED",
IF(DaysOpen__c > 10, "YELLOW",
"GREEN") )
```

Case Data Completeness Tracking

This formula calculates the percentage of specific custom fields that contain data. The formula checks the values of two custom number fields: `Problem Num` and `Severity Num`. If the fields are empty, the formula returns the value "0." The formula returns a value of "1" for each field that contains a value and multiplies this total by fifty to give you the percentage of fields that contain data.

```
(IF(ISBLANK(Problem_Num__c), 0, 1) + IF(ISBLANK(Severity_Num__c ),
0,1)) * 50
```

Suggested Agent Prompts

This formula prompts an agent with cross-sell offers based on past purchases.

```
CASE(Product_Purch__c,
"Printer", "Extra toner cartridges", "Camera", "Memory cards",
"Special of the day")
```

Suggested Offers

This formula suggests a product based on the support history for a computer reseller. When the `Problem` custom field matches a field, the formula field returns a suggestion.

```
CASE(Problem__c,
"Memory", "Suggest new memory cards", "Hard Drive failure", "Suggest
new hard drive with tape backup",
"")
```

Sample Commission Calculations Formulas

For details about using the functions included in these samples, see

Commission Amounts for Opportunities

The following is a simple formula where commission is based on a flat 2% of the opportunity `Amount`.

```
IF(ISPICKVAL(StageName, "Closed Won"),
  ROUND(Amount *0.02, 2), 0)
```

This example calculates the commission amount for any opportunity that has a "Closed Won" stage. The value of this field will be the amount times 0.02 for any closed/won opportunity. Open or lost opportunities will have a zero commission value.

Commission Deal Size

This formula calculates a commission rate based on deal size, returning a 9% commission rate for deals over 100,000 and an 8% commission rate for smaller deals.

```
IF(Amount > 100000, 0.09, 0.08 )
```

Commission Greater Than or Equal To

This formula assigns the "YES" value with a commission greater than or equal to one million. Note, this is a text formula field that uses a custom currency field called `Commission`.

```
IF(Commission__c >=
              1000000, "YES", "NO")
```

Commission Maximum

This formula determines what commission to log for an asset based on which is greater: the user's commission percentage of the price, the price times the discount percent stored for the account or 100 dollars. This example assumes you have two custom percent fields on users and assets.

```
MAX($User.Commission_Percent__c * Price,
  Price * Account_Discount__c, 100)
```

Sample Contact Management Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Contact's Account Discount Percent

This percent formula displays the account's `Discount Percent` field on the contacts page.

```
Account.Discount_Percent__c
```

Contact's Account Name

This formula displays the standard `Account Name` field on the contacts page.

```
Account.Name
```

Contact's Account Phone

This formula displays the standard `Account Phone` field on the contacts page.

```
Account.Phone
```

Contact's Account Rating

Use this formula to display the `Account Rating` field on the contacts page.

```
CASE(Account.Rating, "Hot", "Hot", "Warm", "Warm", "Cold", "Cold",
"Not Rated")
```

Contact's Account Website

This formula displays the standard `Account Website` field on the contacts page.

```
Account.Website
```

If the account website URL is long, use the HYPERLINK function to display a label such as "Click Here" instead of the URL. For example:

```
IF(Account.Website="", "",
 IF(
  OR(LEFT(Account.Website, 7) = "http://",LEFT(Account.Website, 8) =
"https://"),
   HYPERLINK( Account.Website , "Click Here" ),
   HYPERLINK( "http://" & Account.Website , "Click Here" )
 )
)
```

This formula also adds the necessary "http://" or "https://" before a URL if neither were included in the URL field.

Contact's LinkedIn™ Profile

You can configure a link that appears on your contacts' profile page that sends you to their LinkedIn profile. To do so:

1. From the object management settings for contacts, go to Buttons, Links, and Actions.

2. Click **New Button or Link**.

3. Enter a `Label` for this link, like *LinkedInLink*.

4. Enter this formula in the content box:

   ```
   http://www.linkedin.com/search/fpsearch?type=people&keywords
   ={!Contact.FirstName}+{!Contact.LastName}
   ```

5. Click **Save**.

Remember to add this link to the Contact page layout in order for it to show up.

Contact Identification Numbering

This formula displays the first five characters of a name and the last four characters of a social security number separated by a dash. Note that this example uses a text custom field called `SSN`.

```
TRIM(LEFT(LastName, 5)) &
                    "-" & TRIM(RIGHT(SSN__c, 4))
```

6

Contact Preferred Phone

This formula displays the contact's preferred contact method in a contact related list—work phone, home phone, or mobile phone—based on a selected option in a `Preferred Phone` custom picklist.

```
CASE(Preferred_Phone__c,
"Work", "w. " & Phone,
"Home", "h. " & HomePhone,
"Mobile", "m. " & MobilePhone,
"No Preferred Phone")
```

Contact Priority

This formula assesses the importance of a contact based on the account rating and the contact's title. If the account rating is `Hot` or the title starts with `Executive`, then the priority is high (P1). If the account rating is `Warm` or the title starts with `VP` then the priority is medium (P2), and if the account rating is `Cold` then the priority is low (P3).

```
IF(OR(ISPICKVAL(Account.Rating, "Hot"), CONTAINS(Title, "Executive")),
 "P1",

IF(OR(ISPICKVAL(Account.Rating, "Warm"), CONTAINS(Title, "VP")), "P2",


IF(ISPICKVAL(Account.Rating, "Cold"), "P3",

"P3")
)
)
```

Contact Yahoo! ID

This formula displays a clickable Yahoo! Messenger icon indicating if the person is logged on to the service. Users can click the icon to launch a Yahoo! Messenger conversation with the person. This example uses a custom text field called `Yahoo Name` on contacts where you can store the contact's Yahoo! Messenger ID.

```
HYPERLINK("ymsgr:sendIM?" & Yahoo_Name__c,
                   IMAGE("http://opi.yahoo.com/online?u=" &
Yahoo_Name__c &
                   "&m;=g&t;=0", "Yahoo"))
```

Dynamic Address Formatting

This formula field displays a formatted mailing address for a contact in standard format, including spaces and line breaks where appropriate depending on the country.

```
CASE(ShippingCountry,
"USA",
  ShippingStreet & BR() &
  ShippingCity & ",
  " & ShippingState & " " &
```

```
  ShippingPostalCode & BR()
  & ShippingCountry,
"France",
  ShippingStreet & BR() &
  ShippingPostalCode & " " &
  ShippingCity & BR() &
  ShippingCountry, "etc")
```

Telephone Country Code

This formula determines the telephone country code of a contact based on the `Mailing Country` of the mailing address.

```
CASE(MailingCountry,
"USA", "1",
"Canada", "1",
"France", "33",
"UK", "44",
"Australia", "61",
"Japan", "81",
"?")
```

Unformatted Phone Number

This formula removes the parentheses and dash characters from North American phone numbers. This is necessary for some auto-dialer software.

```
IF(Country_Code__c = "1", MID( Phone ,2, 3) & MID(Phone,7,3) &
MID(Phone,11,4), Phone)
```

Sample Data Categorization Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Deal Size Large and Small

This formula displays "Large Deal" for deals over one million dollars or "Small Deal" for deals under one million dollars.

```
IF(Sales_Price__c > 1000000,
"Large Deal",
"Small Deal")
```

Deal Size Small

This formula displays "Small" if the price and quantity are less than one. This field is blank if the asset has a price or quantity greater than one.

```
IF(AND(Price<1,Quantity<1),"Small", null)
```

Product Categorization

This formula checks the content of a custom text field named `Product_Type` and returns "Parts" for any product with the word "part" in it. Otherwise, it returns "Service." Note that the values are case sensitive, so if a `Product_Type` field contains the text "Part" or "PART," this formula returns "Services."

```
IF(CONTAINS(Product_Type__c, "part"), "Parts", "Service")
```

Using Date and Date/Time Values in Formulas

Date formulas are useful for managing payment deadlines, contract ages, or any other features of your organization that are time or date dependent.

Two data types are used for working with dates: Date and Date/Time. Most values that are used when working with dates are of the Date data type, which store the year, month, and day. Some fields, such as `CreatedDate`, are Date/Time fields, meaning they not only store a date value, but also a time value (stored in GMT but displayed in the users' time zone). Date and Date/Time fields are formatted in the user's locale when viewed in reports and record detail pages.

You can use operations like addition and subtraction on Date and Date/Time values to calculate a future date or elapsed time between two dates. If you subtract one date from another, for example, the resulting value will be the difference between the two initial values in days (Number data type). The same operation between two Date/Time values returns a decimal value indicating the difference in number of days, hours, and minutes.

For example, if the difference between two Date/Time values is 5.52, that means the two values are separated by five days, 12 hours (0.5 of a day), and 28 minutes (0.02 of a day). You can also add numeric values to Dates and Date/Times. For example, the operation `TODAY() + 3` returns three days after today's date. For more information and examples of working with dates, see the list of Sample Date Formulas.

Throughout the examples, the variables *date* and *date/time* are used in place of actual Date and Date/Time fields or values.

Keep in mind that complex date functions tend to compile to a larger size than text or number formula functions, so you might run into issues with formula compile size. See *Tips for Reducing Formula Size* for help with this problem.

TODAY() and NOW()

The `TODAY()` function returns the current day, month, and year as a Date data type. This function is useful for formulas where you are concerned with how many days have passed since a previous date, the date of a certain number of days in the future, or if you just want to display the current date.

The `NOW()` function returns the Date/Time value of the current moment. It's useful when you are concerned with specific times of day as well as the date.

For details on how to convert between Date values and Date/Time values, see Converting Between Date/Time and Date.

The DATE() Function

The `DATE()` function returns a Date value, given a year, month, and day. Numerical Y/M/D values and the `YEAR()`, `MONTH()`, and `DAY()` functions are valid parameters for `DATE()`. For example `DATE(`

`2013, 6, 1 )` returns June 6, 2013. Similarly, `DATE( YEAR( TODAY() ), MONTH( TODAY() ) + 3, 1)` returns the Date value of the first day three months from today in the current year, assuming the date is valid (for example, the month falls between 1 and 12).

If the inputted Y/M/D values result in an invalid date, the `DATE()` function returns an error, so error checking is an important part of working with Date values. You can read about methods for handling invalid dates in Sample Date Formulas.

Converting Between Date/Time and Date

Date and Date/Time aren't interchangeable data types, so when you want to perform operations between Date and Date/Time values, you need to convert the values so they are both the same type. Some functions (such as `YEAR()`, `MONTH()`, and `DAY()`) also only work on Date values, so Date/Time values must be converted first.

Use the `DATEVALUE( date/time )` function to return the Date value of a Date/Time. For example, to get the year from a Date/Time, use `YEAR( DATEVALUE( date/time ) ) )`.

You can convert a Date value to a Date/Time using the `DATETIMEVALUE( date )` function. The time will be set to 12:00 a.m. in Greenwich Mean Time (GMT), and then converted to the time zone of the user viewing the record when it's displayed. For a user located in San Francisco, `DATETIMEVALUE( TODAY() )` returns 5:00 p.m. on the previous day (during Daylight Saving Time) rather than 12:00 a.m. of the current day. See A Note About Date/Time and Time Zones for more information.

Converting Between Date and Text

If you want to include a date as part of a string, wrap the Date value in the `TEXT()` function to convert it to text. For example, if you want to return today's date as text, use:

```
"Today's date is " & TEXT( TODAY() )
```

This returns the date in the format "YYYY-MM-DD" rather than in the locale-dependent format. You can change the format by extracting the day, month, and year from the date first and then recombining them in the format you want. For example:

```
"Today's date is " & TEXT( MONTH( date ) ) & "/" & TEXT( DAY( date )
 ) & "/" & TEXT( YEAR( date ) ) )
```

You can also convert text to a Date so you can use the string value with your other Date fields and formulas. You'll want your text to be formatted as "YYYY-MM-DD". Use this formula to return the Date value:

```
DATEVALUE( "YYYY-MM-DD" )
```

Converting Between Date/Time and Text

You can include Date/Time values in a string using the `TEXT()` function, but you need to be careful of time zones. For example, consider this formula:

```
"The current date and time is " & TEXT( NOW() )
```

In this formula, `NOW()` is offset to GMT. Normally, `NOW()` would be converted to the user's time zone when viewed, but because it's been converted to text, the conversion won't happen. So if you execute this formula on August 1st at 5:00 PM in San Francisco time (GMT-7), the result is "The current date and time is 2013–08–02 00:00:00Z".

When you convert a Date/Time to text, a "Z" is included at the end to indicate GMT. `TEXT( ` *date/time* `)` returns "Z" if the field is blank. So if the Date/Time value you're working with might be blank, check for this before converting to text:

```
IF(
  ISBLANK( date/time ),
  "",
  TEXT( date/time )
)
```

To convert a string to a Date/Time value, use `DATETIMEVALUE()` passing in a string in the format "YYYY-MM-DD HH:MM:SS". This method returns the Date/Time value in GMT.

A Note About Date/Time and Time Zones

Date and Date/Time values are stored in GMT. When a record is saved, field values are adjusted from the user's time zone to GMT, and then adjusted back to the viewer's time zone when displayed in record detail pages and reports. With Date conversions this doesn't pose a problem, since converting a Date/Time to a Date results in the same Date value.

When working with Date/Time fields and values, however, the conversion is always done in GMT, not the user's time zone. Subtracting a standard Date/Time field from another isn't a problem because both fields are in the same time zone. When one of the values in the calculation is a conversion from a Text or Date value to a Date/Time value, however, the results are different.

Let's say a San Francisco user enters a value of 12:00 AM on August 2, 2013 in a custom Date/Time field called `Date_Time_c`. This value is stored as 2013–08–02 07:00:00Z, because the time difference in Pacific Daylight Time is GMT-7. At 12:00 p.m. PDT on August 1st, the user views the record and the following formula is run:

```
Date_Time_c - NOW()
```

In the calculation, `NOW()` is 2013–08–01 19:00:00Z, and then subtracted from 2013–08–02 07:00:00Z, to return the expected result of 0.5 (12 hours).

Suppose that instead of NOW(), the formula converts the string "2013–08–01 12:00:00" to a Date/Time value:

```
Date_Time_c - DATETIMEVALUE( "2013-08-01 12:00:00" )
```

In this case, `DATETIMEVALUE( "2013-08-01 12:00:00" )` is 2013–08–01 12:00:00Z, and returns a result of 0.79167, or 19 hours.

There's no way to determine a user's time zone in a formula. If all of your users are in the same time zone, you can adjust the time zone difference by adding or subtracting the time difference between the users' time zone and GMT to your converted values. However, since time zones can be affected by Daylight Saving Time, and the start and end dates for DST are different each year, this is difficult to manage in a formula. We recommend using Apex for transactions that require converting between Date/Time values and Text or Date values.

Sample Date Formulas

Finding the Day, Month, or Year from a Date

Use the functions DAY(*date*), MONTH(*date*), and YEAR(*date*) to return their respective numerical values. Replace *date* with a value of type Date (e.g. TODAY()).

To use these functions with Date/Time values, first convert them to a date with the DATEVALUE() function. For example, DAY(DATEVALUE(*date/time*)).

Finding Out if a Year Is a Leap Year

This formula determines whether or not a year is a leap year. A year is only a leap year if it's divisible by 400, or if it's divisible by four but NOT by 100.

```
OR(
  MOD( YEAR( date ), 400 ) = 0,
  AND(
   MOD( YEAR( date ), 4 ) = 0,
    MOD( YEAR( date ), 100 ) != 0
  )
)
```

Finding Which Quarter a Date Is In

For standard quarters, you can determine which quarter a date falls in using this formula. This formula returns the number of the quarter in which *date* falls (1–4) by dividing the current month by three (the number of months in each quarter) and taking the ceiling.

```
CEILING( MONTH ( date ) / 3 )
```

The formula for shifted quarters is similar, but shifts the month of the date by the number of months between January and the first quarter of the fiscal year. The example below illustrates how you can find a date's quarter if Q1 starts in February instead of January.

```
CEILING( ( MONTH ( date ) - 1 ) / 3)
```

If you want to check whether a date is in the current quarter, add a check to compare the date's year and quarter with TODAY()'s year and quarter.

```
 AND(
   CEILING( MONTH( date ) / 3 ) = CEILING( MONTH( TODAY() ) / 3 ),
   YEAR( date ) = YEAR( TODAY() )
)
```

Finding the Week of the Year a Date Is In

To find the number of a date's week of the year, use this formula:

```
IF(
  CEILING( ( date - DATE( YEAR( date ), 1, 1) + 1) / 7) > 52,
  52,
```

```
  CEILING( ( date - DATE( YEAR( date ), 1, 1) + 1) / 7)
)
```

You can find the current week by determining how many days there have been in the current year and dividing that value by 7. The `IF()` statement ensures that the week number the formula returns doesn't exceed 52. So if the given date is December 31 of the given year, the formula returns 52, even though it's more than 52 weeks after the week of January.

Finding Whether Two Dates Are in the Same Month

To determine whether two Dates fall in the same month, say for a validation rule to determine whether an opportunity Close Date is in the current month, use this formula:

```
AND(
  MONTH( date_1 ) == MONTH( date_2 ),
  YEAR( date_1 ) == YEAR( date_2 )
)
```

Finding the Last Day of the Month

The easiest way to find the last day of a month is to find the first day of the next month and subtract a day.

```
IF(
  MONTH( date ) = 12,
  DATE( YEAR( date ), 12, 31 ),
  DATE( YEAR( date ), MONTH ( date ) + 1, 1 ) - 1
)
```

Displaying the Month as a String Instead of a Number

To return the month as a text string instead of a number, use:

```
CASE(
  MONTH( date ),
  1, "January",
  2, "February",
  3, "March",
  4, "April",
  5, "May",
  6, "June",
  7, "July",
  8, "August",
  9, "September",
  10, "October",
  11, "November",
  "December"
)
```

If your organization uses multiple languages, you can replace the names of the month with a custom label:

```
CASE(
  MONTH( date ),
  1, $Label.Month_of_Year_1,
  2, $Label.Month_of_Year_2,
  3, $Label.Month_of_Year_3,
  4, $Label.Month_of_Year_4,
  5, $Label.Month_of_Year_5,
  6, $Label.Month_of_Year_6,
  7, $Label.Month_of_Year_7,
  8, $Label.Month_of_Year_8,
  9, $Label.Month_of_Year_9,
  10, $Label.Month_of_Year_10,
  11, $Label.Month_of_Year_11,
  $Label.Month_of_Year_12
)
```

Finding and Displaying the Day of the Week From a Date

To find the day of the week from a Date value, use a known Sunday (e.g. January 7, 1900) and subtract it from the date (e.g. `TODAY()`) to get the difference in days. The `MOD()` function finds the remainder of this result when divided by 7 to give the numerical value of the day of the week between 0 (Sunday) and 6 (Saturday). The formula below finds the result and then returns the text name of that day.

```
CASE(
  MOD( date - DATE( 1900, 1, 7 ), 7 ),
  0, "Sunday",
  1, "Monday",
  2, "Tuesday",
  3, "Wednesday",
  4, "Thursday",
  5, "Friday",
  "Saturday"
)
```

Note that this formula only works for dates after 01/07/1900. If you're working with older dates, use the same process with any Sunday prior to your earliest date (e.g. 01/05/1800).

You can also adjust this formula if your week starts on a different day. For example, if your week starts on Monday, you can use January 8, 1900 in your condition. The new formula looks like this:

```
CASE(
  MOD( date - DATE( 1900, 1, 8 ), 7 ),
  0, "Monday",
  1, "Tuesday",
  2, "Wednesday",
  3, "Thursday",
  4, "Friday",
  5, "Saturday",
  "Sunday"
)
```

Like the formula for getting the name of the month, if your organization uses multiple languages, you can replace the names of the day of the week with a variable like `$Label.Day_of_Week_1`, etc.

Finding the Next Day of the Week After a Date

To find the date of the next occurrence of a particular day of the week following a given Date, get the difference in the number of days of the week between a *date* and a *day_of_week*, a number 0–6 where 0 = Sunday and 6 = Saturday. By adding this difference to the current date, you'll find the date of the *day_of_week*. The `IF()` statement in this formula handles cases where the *day_of_week* is prior to the day of the week of the *date* value (e.g. *date* is a Thursday and *day_of_week* is a Monday) by adding 7 to the difference.

```
date + ( day_of_week - MOD( date - DATE( 1900, 1, 7 ), 7 ) )
+
IF(
  MOD( date - DATE( 1900, 1, 7 ), 7 ) >= day_of_week,
  7,
  0
)
```

You can substitute either a constant or another field in for the *day_of_week* value based on your needs.

Finding the Number of Days Between Two Dates

To find the number of days between two dates, *date_1* and *date_2*, subtract the earlier date from the later date: **date_1 − date_2**

You can alter this slightly if you want to determine a date a certain number of days in the past. For example, say you want a formula to return true if some date field is more than 30 days prior to the current date and false otherwise. This formula does just that:

```
TODAY() - 30 > date
```

Finding the Number of Business Days Between Two Dates

Calculating how many business days passed between two dates is slightly more complex than calculating total elapsed days. The basic strategy is to choose a reference Monday from the past and find out how many full weeks and any additional portion of a week have passed between the reference date and the date you're examining. These values are multiplied by five (for a five-day work week) and then the difference between them is taken to calculate business days.

```
(5 * ( FLOOR( ( date_1 - DATE( 1900, 1, 8) ) / 7 ) ) + MIN( 5, MOD(
date_1 - DATE( 1900, 1, 8), 7 ) ) )
-
(5 * ( FLOOR( ( date_2 - DATE( 1900, 1, 8) ) / 7 ) ) + MIN( 5, MOD(
date_2 - DATE( 1900, 1, 8), 7 ) ) )
```

In this formula, *date_1* is the more recent date and *date_2* is the earlier date. If your work week runs shorter or longer than five days, replace all fives in the formula with the length of your week.

Adding Days, Months, and Years to a Date

If you want to add a certain number of days to a date, add that number to the date directly. For example, to add five days to a date, the formula is **date** + 5.

Adding years to a date is fairly simple, but you do need to check that the future date is valid. That is, adding five years to February 29 (a leap year) results in an invalid date. The following formula adds *num_years* to *date* by checking if the date is February 29 and if the future date is not in a leap year. If these conditions hold true, the formula returns March 1 in the future year. Otherwise, the formula sets the Date to the same month and day *num_years* in the future.

```
IF(
  AND(
    MONTH( date ) = 2,
    DAY( date ) = 29,
    NOT(
      OR(
        MOD( YEAR( date ), 400 ) = 0,
        AND(
          MOD( YEAR( date ), 4 ) = 0,
          MOD( YEAR( date ), 100 ) != 0
        )
      )
    )
  ),
  DATE( YEAR( date ) + num_years, 3, 1),
  DATE( YEAR( date ) + num_years, MONTH( date ), DAY( date ) )
)
```

Adding months to a date is slightly more complicated as months vary in length and the cycle of months restart with each year. Therefore, a valid day in one month (January 31) might not be valid in another month (February 31). A simple solution is to approximate each month's length as 365/12 days:

```
date + ( ( 365 / 12 ) * Number_months )
```

While this formula is a good estimate, it doesn't return an exact date. For example, if you add two months to April 30 using this method, the formula will return June 29 instead of June 30. Returning an exact date depends on your organization's preference. For example, when you add one month to January 31, should it return February 28 (the last day of the next month) or March 2 (30 days after January 31)?

This formula does the following:

* Returns March 1 if the future month is a February and the day is greater than 28. This portion of the formula performs the same for both leap and non-leap years.
* Returns the first day of the next month if the future month is April, June, September, or November and the day is greater than 30.
* Otherwise, it returns the correct date in the future month.

This example formula adds two months to a given date. You can modify the conditions on this formula if you prefer different behaviors for dates at the end of the month.

```
DATE(
  YEAR( date ) + FLOOR( ( MONTH ( date ) + 2 - 1 ) / 12 ),
  MOD( MONTH ( date ) + 2 - 1 +
    IF( DAY ( date ) > CASE( MOD( MONTH( date ) + 2 - 1, 12 ) + 1,
```

```
         2, 28,
         4, 30,
         6, 30,
         9, 30,
         11, 30,
         31 ), 1, 0 ), 12 ) + 1,
      IF( DAY( date ) > CASE( MOD( MONTH( date ) + 2 - 1, 12 ) + 1,
         2, 28,
         4, 30,
         6, 30,
         9, 30,
         11, 30,
         31 ),
      1, DAY( date )
   )
)
```

If you're using these formulas for expiration dates, you might want to subtract a day from the return value to make sure that some action is completed *before* the calculated date.

Adding Business Days to a Date

This formula finds three business days from a given `date`.

```
CASE(
  MOD( date - DATE( 1900, 1, 7 ), 7 ),
  3, date + 2 + 3,
  4, date + 2 + 3,
  5, date + 2 + 3,
  6, date + 1 + 3,
  date + 3
)
```

This formula finds the day of the week of the `date` field value. If the date is a Wednesday, Thursday, or Friday, the formula adds five calendar days (two weekend days, three weekdays) to the date to account for the weekend. If `date` is a Saturday, you need four additional calendar days. For any other day of the week (Sunday — Tuesday), simply add three days. You can easily modify this formula to add more or less business days. The tip for getting the day of the week might be useful if you need to adjust this formula.

Finding the Hour, Minute, or Second from a Date/Time

To get the hour, minute, and second from a Date/Time field as a numerical value, use the following formulas where `TZoffset` is the difference between the user's time zone and GMT. For hour in 24–hour format:

```
VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) )
```

For hour in 12–hour format:

```
IF(
  OR(
    VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) = 0,
    VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) = 12
  ),
```

```
    12,
    VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) )
     -
     IF(
       VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) < 12,
       0,
       12
     )
)
```

For minutes:

```
VALUE( MID( TEXT( date/time - TZoffset ), 15, 2 ) )
```

For seconds:

```
VALUE( MID( TEXT( date/time - TZoffset ), 18, 2 ) )
```

And, to get "AM" or "PM" as a string, use:

```
IF(
  VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) < 12,
  "AM",
  "PM"
)
```

To return the time as a string in "HH:MM:SS A/PM" format, use the following formula:

```
IF(
  OR(
    VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) = 0,
    VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) = 12
  ),
  "12",
  TEXT( VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) )
   -
   IF(
     VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) < 12,
     0,
     12
   )
  )
)
& ":" &
MID( TEXT( date/time - TZoffset ), 15, 2 )
& ":" &
MID( TEXT( date/time - TZoffset ), 18, 2 )
& " " &
IF(
  VALUE( MID( TEXT( date/time - TZoffset ), 12, 2 ) ) < 12,
  "AM",
  "PM"
)
```

When working with time in formula fields, you need to consider the time difference between your organization and GMT. See A Note About Date/Time and Time Zones on page 11 for help understanding the time zone offset used in this formula.

Finding the Elapsed Time Between Date/Times

To find the difference between two Date values as a number, subtract one from the other like so: **date_1 — date_2** to return the difference in days.

Finding the elapsed time between two Date/Time values is slightly more complex. This formula converts the difference between two Date/Time values, **datetime_1** and **datetime_2**, to days, hours, and minutes.

```
IF(
  datetime_1 - datetime_2 > 0 ,
  TEXT( FLOOR( datetime_1 - datetime_2 ) ) & " days "
  & TEXT( FLOOR( MOD( (datetime_1 - datetime_2 ) * 24, 24 ) ) ) & "
hours "
  & TEXT( ROUND( MOD( (datetime_1 - datetime_2 ) * 24 * 60, 60 ), 0
) ) & " minutes",
  ""
)
```

Finding the Number of Business Hours Between Two Date/Times

The formula for finding business hours between two Date/Time values expands on the formula for finding elapsed business days. It works on the same principle of using a reference Date/Time, in this case 1/8/1900 at 16:00 GMT (9 a.m. PDT), and then finding your Dates' respective distances from that reference. The formula rounds the value it finds to the nearest hour and assumes an 8–hour, 9 a.m. – 5 p.m. work day.

```
ROUND( 8 * (
   ( 5 * FLOOR( ( DATEVALUE( date/time_1 ) - DATE( 1900, 1, 8) ) /
7) +
   MIN(5,
    MOD( DATEVALUE( date/time_1 ) - DATE( 1900, 1, 8), 7) +
    MIN( 1, 24 / 8 * ( MOD( date/time_1 - DATETIMEVALUE( '1900-01-08
 16:00:00' ), 1 ) ) )
   )
  )
 -
   ( 5 * FLOOR( ( DATEVALUE( date/time_2 ) - DATE( 1900, 1, 8) ) /
7) +
    MIN( 5,
     MOD( DATEVALUE( date/time_2 ) - DATE( 1996, 1, 1), 7 ) +
    MIN( 1, 24 / 8 * ( MOD( date/time_2 - DATETIMEVALUE( '1900-01-08
 16:00:00' ), 1) ) )
   )
  )
 ),
0 )
```

You can change the eights in the formula to account for a longer or shorter work day. If you live in a different time zone or your work day doesn't start at 9:00 a.m., change the reference time to the start of your work day in GMT. See A Note About Date/Time and Time Zones for more information.

Sample Discounting Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Maintenance and Services Discount

This formula field uses two custom currency fields: `Maintenance Amount` and `Services Amount`. It displays "Discounted" on an opportunity if its maintenance amount and services amount do not equal the opportunity `Amount` standard field value. Otherwise, it displays "Full Price."

```
IF(Maintenance_Amount__c + Services_Amount__c <> Amount,
"Discounted",
"Full Price")
```

Opportunity Discount Amount

This formula calculates the difference of the product `Amount` less the `Discount Amount`. Note that `Discount Amount` is a custom currency field.

```
Amount -
                  Discount_Amount__c
```

Opportunity Discount Rounded

Use this formula to calculate the discounted amount of an opportunity rounded off to two digits. This example is a number formula field on opportunities that uses a custom percent field called `Discount Percent`.

```
ROUND(Amount-Amount* Discount_Percent__c,2)
```

Opportunity Discount with Approval

This formula adds a "Discount Approved" checkbox to an opportunity. It uses conditional logic to check the value of the approval flag before calculating the commission.

```
IF(Discount_Approved__c, ROUND(Amount - Amount * DiscountPercent__c,
2), Amount)
```

Sample Employee Services Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Bonus Calculation

This example determines an employee's bonus amount based on the smallest of two amounts: the employee's gross times bonus percent or an equally divided amount of the company's performance amount among all employees. It assumes you have custom number field for `Number of Employees`, a custom percent field for `Bonus Percent`, and currency custom fields for the employee's `Gross` and company's `Performance`.

```
MIN(Gross__c * Bonus_Percent__c,
  Performance__c / Number_of_Employees__c)
```

Employee 401K

This example formula determines which amount to provide in employee 401K matching based on a matching program of half of the employee's contribution or $250, whichever is less. It assumes you have custom currency field for `Contribution`.

```
MIN(250, Contribution__c /2)
```

Hours Worked Per Week

This formula uses a custom tab to enable time tracking of hours worked per day. It uses a formula field to sum the hours per week.

```
MonHours__c + TuesHours__c + WedsHours__c + ThursHours__c + FriHours__c
```

Total Pay Amount

This formula determines total pay by calculating regular hours multiplied by a regular pay rate, plus overtime hours multiplied by an overtime pay rate.

```
Total Pay =
IF(Total_Hours__c <= 40, Total_Hours__c * Hourly_Rate__c,
40 * Hourly_Rate__c +
(Total_Hours__c - 40) * Overtime_Rate__c)
```

Sample Expense Tracking Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Expense Identifier

This formula displays the text "Expense-" followed by trip name and the expense number. This is a text formula field that uses an expense number custom field.

```
"Expense-" &
                    Trip_Name__c & "-" & ExpenseNum__c
```

Mileage Calculation

This formula calculates mileage expenses for visiting a customer site at 35 cents a mile.

```
Miles_Driven__c * 0.35
```

Sample Financial Calculations Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Compound Interest

This formula calculates the interest, you will have after T years, compounded M times per year.

```
Principal__c * ( 1 + Rate__c / M ) ^ ( T * M) )
```

Compound Interest Continuous

This formula calculates the interest that will have accumulated after T years, if continuously compounded.

```
Principal__c * EXP(Rate__c * T)
```

Consultant Cost

This formula calculates the number of consulting days times 1200 given that this formula field is a currency data type and consulting charges a rate of $1200 per day. Note that `Consulting Days` is a custom field.

```
Consulting_Days__c *
                1200
```

Gross Margin

This formula provides a simple calculation of gross margin. In this formula example, `Total Sales` and `Cost of Goods Sold` are custom currency fields.

```
Total_Sales__c - Cost_of_Goods_Sold__c
```

Gross Margin Percent

This formula calculates the gross margin based on a margin percent.

```
Margin_percent__c * Items_Sold__c * Price_item__c
```

Payment Due Indicator

This formula returns the date five days after the contract start date whenever `Payment Due Date` is blank. `Payment Due Date` is a custom date field.

```
(BLANKVALUE(Payment_Due_Date__c, StartDate +5)
```

Payment Status

This formula determines if the payment due date is past and the payment status is "UNPAID." If so, it returns the text "PAYMENT OVERDUE" and if not, it leaves the field blank. This example uses a custom date field called `Payment Due Date` and a text custom field called `Payment Status` on contracts.

```
IF(
AND(Payment_Due_Date__c < TODAY(),
ISPICKVAL(Payment_Status__c, "UNPAID")),
"PAYMENT OVERDUE",
null )
```

Sample Image Link Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Yahoo! Instant Messenger™ Image

This formula displays an image that indicates whether a contact or user is currently logged in to Yahoo! Instant Messenger. Clicking the image launches the Yahoo! Instant Messenger window. This formula uses a custom text field called `Yahoo Name` to store the contact or user's Yahoo! ID.

```
IF(ISBLANK(Yahoo_Name__c),"", HYPERLINK("ymsgr:sendIM?" &
Yahoo_Name__c,
IMAGE("http://opi.yahoo.com/online?u=" & Yahoo_Name__c & "&m=g&t=0",
" ")))
```

Flags for Case Priority

This formula displays a green, yellow, or red flag image to indicate case priority.

```
IMAGE(
CASE( Priority,
"Low", "/img/samples/flag_green.gif",
"Medium", "/img/samples/flag_yellow.gif",
"High", "/img/samples/flag_red.gif",
"/s.gif"),
"Priority Flag")
```

23

Color Squares for Case Age

This formula displays a 30 x 30 pixel image of a red, yellow, or green, depending on the value of a `Case Age` custom number field.

```
IF( Case_Age__c > 20,
IMAGE("/img/samples/color_red.gif", "red", 30, 30),
IF( Case_Age__c > 10,
IMAGE("/img/samples/color_yellow.gif", "yellow", 30, 30),
IMAGE("/img/samples/color_green.gif", "green", 30, 30)
))
```

Traffic Lights for Status

This formula displays a green, yellow, or red traffic light images to indicate status, using a custom picklist field called `Project Status`. Use this formula in list views and reports to create a "Status Summary" dashboard view.

```
IMAGE(
CASE(Project_Status__c,
"Green", "/img/samples/light_green.gif",
"Yellow", "/img/samples/light_yellow.gif",
"Red", "/img/samples/light_red.gif",
"/s.gif"),
"status color")
```

Stars for Ratings

This formula displays a set of one to five stars to indicate a rating or score.

```
IMAGE(
CASE(Rating__c,
"1", "/img/samples/stars_100.gif",
"2", "/img/samples/stars_200.gif",
"3", "/img/samples/stars_300.gif",
"4", "/img/samples/stars_400.gif",
"5", "/img/samples/stars_500.gif",
"/img/samples/stars_000.gif"),
"rating")
```

Consumer Reports™—Style Colored Circles for Ratings

This formula displays a colored circle to indicate a rating on a scale of one to five, where solid red is one, half red is two, black outline is three, half black is four, and solid black is five.

```
IMAGE(
CASE(Rating__c,
"1", "/img/samples/rating1.gif",
"2", "/img/samples/rating2.gif",
"3", "/img/samples/rating3.gif",
"4", "/img/samples/rating4.gif",
"5", "/img/samples/rating5.gif",
```

```
"/s.gif"),
"rating")
```

Horizontal Bars to Indicate Scoring

This formula displays a horizontal color bar (green on a white background) of a length that is proportional to a numeric score. In this example, the maximum length of the bar is 200 pixels.

```
IMAGE("/img/samples/color_green.gif", "green", 15, Industry_Score__c
* 2) &
IMAGE("/s.gif", "white", 15,
200 - (Industry_Score__c * 2))
```

Sample Integration Link Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Application API Link

This formula creates a link to an application outside Salesforce, passing the parameters so that it can connect to Salesforce via the SOAP API and create the necessary event.

```
HYPERLINK ("https://www.myintegration.com?sId=" & GETSESSIONID() &
"?&rowID=" & Name & "action=CreateTask","Create a Meeting Request")
```

🛑 Important: `$Api.Session_ID` and `GETSESSIONID()` return the same value, an identifier for the current session in the current context. This context varies depending on where the global variable or function is evaluated. For example, if you use either in a custom formula field, and that field is displayed on a standard page layout in Salesforce Classic, the referenced session will be a basic Salesforce session. That same field (or the underlying variable or formula result), when used in a Visualforce page, references a Visualforce session instead.

Session contexts are based on the domain of the request. That is, the session context changes whenever you cross a hostname boundary, such as from `.salesforce.com` to `.visual.force.com` or `.lightning.force.com`.

Session identifiers from different contexts, and the sessions themselves, are different. When you transition between contexts, the old session is replaced by the new one, and the old session is no longer valid. The session ID also changes at this time.

Normally Salesforce transparently handles session hand-off between contexts, but if you're passing the session ID around yourself, be aware that you might need to re-access `$Api.Session_ID` or `GETSESSIONID()` from the new context to ensure a valid session ID.

Note also that not all sessions are created equal. In particular, sessions obtained in a Lightning Experience context have reduced privileges, and don't have API access. You can't use these session IDs to make API calls.

Shipment Tracking Integration

This formula creates a link to FedEx, UPS, or DHL shipment tracking websites, depending on the value of a `Shipping Method` custom picklist field. Note that the parameters shown in this example for FedEx, UPS, and DHL websites are illustrative and do not represent the correct parameters for all situations.

```
CASE(Shipping_Method__c,
"Fedex",
HYPERLINK("http://www.fedex.com/Tracking?ascend_header=1&clienttype
=dotcom&cntry_code=us&language=english&tracknumbers= "&
tracking_id__c,"Track"),
"UPS",
HYPERLINK("http://wwwapps.ups.com/WebTracking/processInputRequest?HTMLVersion
=5.0&sort_by=status&loc=en_US&InquiryNumber1= "& tracking_id__c &
"&track.x=32&track.y=7", "Track") ,
"DHL",
HYPERLINK("http://track.dhl-usa.com/TrackByNbr.asp?ShipmentNumber=" &
 tracking_id__c,"Track"), "")
```

Skype™ Auto Dialer Integration

This formula creates a linkable phone number field that automatically dials the phone number via the Skype VOIP phone application. It requires installation of the Skype application (a third-party product not provided by Salesforce) on your desktop.

```
HYPERLINK("callto://+" & Country_Code__c & Phone_Unformatted__c, Phone)
```

Sample Lead Management Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Lead Aging (for open leads)

This formula checks to see if a lead is open and if so, calculates the number of days it has been open by subtracting the date and time created from the current date and time. The result is the number of days open rounded to zero decimal places. If the lead is not open, this field is blank.

```
IF(ISPICKVAL(Status,
                   "Open"), ROUND(NOW()-CreatedDate, 0), null)
```

Lead Data Completeness

This formula calculates the percent of certain lead fields that your sales personnel enter. The formula field checks the values of two custom number fields: `Phone` and `Email`. If the fields are empty, the formula returns the value "0." The formula returns a value of "1" for each field that contains a value and multiplies this total by fifty to give you the percentage of fields that contain data.

```
(IF(Phone = "", 0, 1) + IF(Email = "", 0, 1) ) * 50
```

Lead Numbering

This formula returns a number value for the text value in the auto-number field `Lead Number`. This can be useful if you want to use the `Lead Number` field in a calculation, such as round-robin or other routing purposes. Note that auto-number fields are text fields and must be converted to a number for numeric calculations.

```
VALUE(Lead_Number__c)
```

Round-Robin Assignment of Cases or Leads

The following formula example for leads assumes you have three lead queues and you want to assign an equal number of incoming leads to each queue. You can also assign cases using a similar formula.

```
MOD(VALUE(Lead_Number__c),
               3)
```

This formula is for a custom formula field named Round_Robin_ID that assigns each lead a value of 0, 1, or 2. This formula uses a custom auto-number field called `Lead Number` that assigns each lead a unique number starting with 1. The MOD function divides the lead number by the number of lead queues available (three in this example) and returns a remainder of 0, 1, or 2. Use the value of this formula field in your lead assignment rules to assign lead records to different queues. For example:

- Round_Robin_ID = 0 is assigned to Queue A
- Round_Robin_ID = 1 is assigned to Queue B
- Round_Robin_ID = 2 is assigned to Queue C

Sample Metrics Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Temperature Conversion

This formula converts Celsius degrees to Fahrenheit.

```
1.8 * degrees_celsius__c + 32
```

Unit of Measure Conversion

This formula converts kilometers to miles.

```
Miles__c/.621371192
```

Sample Opportunity Management Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Expected Product Revenue

This formula calculates total revenue from multiple products, each with a different probability of closing.

```
ProductA_probability__c * ProductA_revenue__c + ProductB_probability__c
 * ProductB_revenue__c
```

Maintenance Calculation

This formula calculates maintenance fees as 20% of license fees per year. `Maintenance Years` is a custom field on opportunities.

```
Amount * Maint_Years__c * 0.2
```

Monthly Subscription-Based Calculated Amounts

This formula calculates an opportunity amount based on a monthly subscription rate multiplied by the subscription period.

```
Monthly_Amount__c * Subscription_Months__c
```

Monthly Value

This formula divides total yearly value by 12 months.

```
Total_value__c / 12
```

Opportunity Additional Costs

This formula calculates the sum of the product `Amount`, maintenance amount, and services fees. Note that `Maint amount` and `Service Fees` are custom currency fields.

```
Amount + Maint_Amount__c +
                 Services_Amount__c
```

Opportunity Categorization

This formula uses conditional logic to populate an `Opportunity category` text field, based on the value of the `Amount` standard field. Opportunities with amounts less than $1500 are "Category 1," those between $1500 and $10000 are "Category 2," and the rest are "Category 3." This example uses nested IF statements.

```
IF(Amount < 1500, "Category 1", IF(Amount > 10000, "Category 3",
"Category 2"))
```

Opportunity Data Completeness

This formula takes a group of fields and calculates what percent of them are being used by your personnel. This formula field checks five fields to see if they are blank. If so, a zero is counted for that field. A "1" is counted for any field that contains a value and this total is divided by five (the number of fields evaluated).

Note that this formula requires you select the `Treat blank fields as blanks` option under Blank Field Handling while the Advanced Formula subtab is showing.

```
(IF(ISBLANK(Maint_Amount__c), 0, 1) +
 IF(ISBLANK(Services_Amount__c), 0,1) +
  IF(ISBLANK(Discount_Percent__c), 0, 1) +
   IF(ISBLANK(Amount), 0, 1) +
    IF(ISBLANK(Timeline__c), 0, 1)) / 5
```

Opportunity Expected License Revenue

This formula calculates expected revenue for licenses based on probability of closing.

```
Expected_rev_licenses__c * Probability
```

Opportunity Revenue Text Display

This formula returns the expected revenue amount of an opportunity in text format without a dollar sign. For example, if the `Expected Revenue` of a campaign is "$200,000," this formula field displays "200000."

```
TEXT(ExpectedRevenue)
```

Opportunity Total Deal Size

This formula calculates the sum of maintenance and services amounts.

```
Amount + Maint_Amount__c + Services_Amount__c
```

Opportunity Total Price Based on Units

This formula generates proposal pricing based on unit price and total volume.

```
Unit_price__c * Volume__c * 20
```

Professional Services Calculation

This formula estimates professional service fees at an average loaded rate of $1200 per day. `Consulting Days` is a custom field on opportunities.

```
Consulting_Days__c * 1200
```

Stage-Based Sales Document Selection

This formula Identifies a relevant document in the Documents tab based on opportunity `Stage`. Use document IDs in the form of "00l30000000j7AO."

```
CASE(StageName,
"Prospecting", "Insert 1st Document ID",
"Qualification", "Insert 2nd Document ID",
"Needs Analysis", "Insert 3rd Document ID",
```

```
"Value Proposition", ...
)
)
```

Sales Coach

This formula creates a hyperlink that opens a stage-specific document stored in the Documents tab. It uses the previously defined custom formula field that identifies a document based on opportunity `Stage`. See Stage-Based Sales Document Selection on page 29.

```
HYPERLINK("/servlet/servlet.FileDownload?file=" & Relevant_Document__c,
 "View Document in New Window")
```

Shipping Cost by Weight

This formula calculates postal charges based on weight.

```
package_weight__c * cost_lb__c
```

Shipping Cost Percentage

This formula calculates shipping cost as a fraction of `total amount`.

```
Ship_cost__c / total_amount__c
```

Tiered Commission Rates

This formula calculates the 2% commission amount of an opportunity that has a probability of 100%. All other opportunities will have a commission value of zero.

```
IF(Probability = 1,
ROUND(Amount * 0.02, 2),
0)
```

Total Contract Value from Recurring and Non-Recurring Revenue

This formula calculates both recurring and non-recurring revenue streams over the lifetime of a contract.

```
Non_Recurring_Revenue__c + Contract_Length_Months__c *
Recurring_Revenue__c
```

Sample Pricing Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Total Amount

This formula calculates a total amount based on unit pricing and total units.

```
Unit_price__c * Total_units__c
```

User Pricing

This formula calculates a price per user license.

```
Total_license_rev__c / Number_user_licenses__c
```

Sample Scoring Calculations Formulas

For details about using the functions included in these samples, see Formula Operators and Functions on page 31.

Lead Scoring

This formula scores leads, providing a higher score for phone calls than website requests.

```
CASE(LeadSource, "Phone", 2, "Web", 1, 0)
```

Here's a formula that scores a lead based on his or her rating:

```
CASE(1, IF(ISPICKVAL(Rating, "Hot"), 1, 0), 3, IF(ISPICKVAL(Rating,
"Warm"), 1, 0), 2, IF(ISPICKVAL(Rating, "Cold"), 1, 0), 1))
```

Customer Success Scoring

This formula uses a simple scoring algorithm to rank customers a high score for positive survey results in Salesforce.

```
Survey_Question_1__c * 5 + Survey_Question_2__c *2
```

Formula Operators and Functions

Use these operators and functions when building formulas. All functions are available everywhere that you can include a formula—such as formula fields, validation rules, approval processes, and workflow rules, unless otherwise specified.

Note: Extraneous spaces in the samples below are ignored.

- Math Operators
- Logical Operators
- Text Operators
- Date and Time Functions
- Informational Functions
- Logical Functions
- Math Functions

- Text Functions
- Summary Functions
- Advanced Functions
- Encoding Functions

Math Operators

Operator	Description
+ (Add)	Calculates the sum of two values.
- (Subtract)	Calculates the difference of two values.
* (Multiply)	Multiplies its values.
/ (Divide)	Divides its values.
^ (Exponentiation)	Raises a number to a power of a specified number.
() (Open Parenthesis and Close Parenthesis)	Specifies that the expressions within the open parenthesis and close parenthesis are evaluated first. All other expressions are evaluated using standard operator precedence.

Logical Operators

Operator	Description
= and == (Equal)	Evaluates if two values are equivalent. The = and == operator are interchangeable.
<> and != (Not Equal)	Evaluates if two values are not equivalent.
< (Less Than)	Evaluates if a value is less than the value that follows this symbol.
> (Greater Than)	Evaluates if a value is greater than the value that follows this symbol.
<= (Less Than or Equal)	Evaluates if a value is less than or equal to the value that follows this symbol.
>= (Greater Than or Equal)	Evaluates if a value is greater than or equal to the value that follows this symbol.
&& (AND)	Evaluates if two values or expressions are both true. Use this operator as an alternative to the logical function AND.
|| (OR)	Evaluates if at least one of multiple values or expressions is true. Use this operator as an alternative to the logical function OR.

Text Operators

Operator	Description
& (Concatenate)	Connects two or more strings.

Date and Time Functions

Function	Description
DATE	Returns a date value from year, month, and day values you enter. Salesforce displays an error on the detail page if the value of the DATE function in a formula field is an invalid date, such as February 29 in a non-leap year.
DATEVALUE	Returns a date value for a date/time or text expression.
DATETIMEVALUE	Returns a year, month, day and GMT time value.
DAY	Returns a day of the month in the form of a number between 1 and 31.
MONTH	Returns the month, a number between 1 (January) and 12 (December) in number format of a given date.
NOW	Returns a date/time representing the current moment.
TODAY	Returns the current date as a date data type.
YEAR	Returns the four-digit year in number format of a given date.

Informational Functions

Function	Description
BLANKVALUE	Determines if an expression has a value and returns a substitute expression if it does not. If the expression has a value, returns the value of the expression.
ISBLANK	Determines if an expression has a value and returns TRUE if it does not. If it contains a value, this function returns FALSE.
ISNULL	Determines if an expression is null (blank) and returns TRUE if it is. If it contains a value, this function returns FALSE. ⛔ **Important:** Use ISBLANK instead of ISNULL in new formulas. ISBLANK has the same functionality as ISNULL, but also supports text fields. Salesforce will continue to support ISNULL, so you do not need to change any existing formulas.
NULLVALUE	Determines if an expression is null (blank) and returns a substitute expression if it is. If the expression is not blank, returns the value of the expression. ⛔ **Important:** Use BLANKVALUE instead of NULLVALUE in new formulas. BLANKVALUE has the same functionality as NULLVALUE, but also supports

Function	Description
	text fields. Salesforce will continue to support NULLVALUE, so you do not need to change existing formulas.
PRIORVALUE	Returns the previous value of a field.

Logical Functions

Function	Description
AND	Returns a TRUE response if all values are true; returns a FALSE response if one or more values are false.
CASE	Checks a given expression against a series of values. If the expression is equal to a value, returns the corresponding result. If it is not equal to any values, it returns the `else_result`.
IF	Determines if expressions are true or false. Returns a given value if true and another value if false.
ISCHANGED	Compares the value of a field to the previous value and returns TRUE if the values are different. If the values are the same, this function returns FALSE.
ISNEW	Checks if the formula is running during the creation of a new record and returns TRUE if it is. If an existing record is being updated, this function returns FALSE.
ISNUMBER	Determines if a text value is a number and returns TRUE if it is. Otherwise, it returns FALSE.
NOT	Returns FALSE for TRUE and TRUE for FALSE.
OR	Determines if expressions are true or false. Returns TRUE if any expression is true. Returns FALSE if all expressions are false.

Math Functions

Function	Description
ABS	Calculates the absolute value of a number. The absolute value of a number is the number without its positive or negative sign.
CEILING	Rounds a number up to the nearest integer.
DISTANCE	Calculates the distance between two locations in miles or kilometers.
EXP	Returns a value for e raised to the power of a number you specify.
FLOOR	Returns a number rounded down to the nearest integer.
GEOLOCATION	Returns a geolocation based on the provided latitude and longitude. Must be used with the DISTANCE function.

Function	Description
LN	Returns the natural logarithm of a specified number. Natural logarithms are based on the constant e value of 2.71828182845904.
LOG	Returns the base 10 logarithm of a number.
MAX	Returns the highest number from a list of numbers.
MIN	Returns the lowest number from a list of numbers.
MOD	Returns a remainder after a number is divided by a specified divisor.
ROUND	Returns the nearest number to a number you specify, constraining the new number by a specified number of digits.
SQRT	Returns the positive square root of a given number.

Text Functions

Function	Description
BEGINS	Determines if text begins with specific characters and returns TRUE if it does. Returns FALSE if it does not.
BR	Inserts a line break in a string of text.
CASESAFEID	Converts a 15-character ID to a case-insensitive 18-character ID.
CONTAINS	Compares two arguments of text and returns TRUE if the first argument contains the second argument. If not, returns FALSE.
FIND	Returns the position of a string within a string of text represented as a number.
GETSESSIONID	Returns the user's session ID.
HYPERLINK	Creates a link to a URL specified that is linkable from the text specified.
IMAGE	Inserts an image with alternate text and height/width specifications.
INCLUDES	Determines if any value selected in a multi-select picklist field equals a text literal you specify.
ISPICKVAL	Determines if the value of a picklist field is equal to a text literal you specify.
LEFT	Returns the specified number of characters from the beginning of a text string.
LEN	Returns the number of characters in a specified text string.
LOWER	Converts all letters in the specified text string to lowercase. Any characters that are not letters are unaffected by this function. Locale rules are applied if a locale is provided.
LPAD	Inserts characters you specify to the left-side of a text string.
MID	Returns the specified number of characters from the middle of a text string given the starting position.

35

Function	Description
RIGHT	Returns the specified number of characters from the end of a text string.
RPAD	Inserts characters that you specify to the right-side of a text string.
SUBSTITUTE	Substitutes new text for old text in a text string.
TEXT	Converts a percent, number, date, date/time, or currency type field into text anywhere formulas are used. Also, converts picklist values to text in approval rules, approval step rules, workflow rules, escalation rules, assignment rules, auto-response rules, validation rules, formula fields, field updates, and custom buttons and links.
TRIM	Removes the spaces and tabs from the beginning and end of a text string.
UPPER	Converts all letters in the specified text string to uppercase. Any characters that are not letters are unaffected by this function. Locale rules are applied if a locale is provided.
VALUE	Converts a text string to a number.

Summary Functions

The following functions are available with summary, matrix, and joined reports.

Function	Description
PARENTGROUPVAL	This function returns the value of a specified parent grouping. A "parent" grouping is any level above the one containing the formula. You can only use this function in custom summary formulas for reports.
PREVGROUPVAL	This function returns the value of a specified previous grouping. A "previous" grouping is one that comes before the current grouping in the report. Choose the grouping level and increment. The increment is the number of columns or rows before the current summary. The default is 1; the maximum is 12. You can only use this function in custom summary formulas for reports.

Advanced Functions

Function	Description
GETRECORDIDS	Returns an array of strings in the form of record IDs for the selected records in a list, such as a list view or related list.
INCLUDE	Returns content from an s-control snippet. Use this function to reuse common code in many s-controls.
LINKTO	Returns a relative URL in the form of a link (href and anchor tags) for a custom s-control or Salesforce page.

Function	Description
REGEX	Compares a text field to a regular expression and returns TRUE if there is a match. Otherwise, it returns FALSE. A regular expression is a string used to describe a format of a string according to certain syntax rules.
REQUIRESCRIPT	Returns a script tag with source for a URL you specify. Use this function when referencing the Force.com AJAX Toolkit or other JavaScript toolkits.
URLFOR	Returns a relative URL for an action, s-control, Visualforce page, or a file in a static resource archive in a Visualforce page.
VLOOKUP	Returns a value by looking up a related value on a custom object similar to the VLOOKUP() Excel function.

Encoding Functions

Function	Description
HTMLENCODE	Encodes text and merge field values for use in HTML by replacing characters that are reserved in HTML, such as the greater-than sign (>), with HTML entity equivalents, such as `>`.
JSENCODE	Encodes text and merge field values for use in JavaScript by inserting escape characters, such as a backslash (\), before unsafe JavaScript characters, such as the apostrophe (').
JSINHTMLENCODE	Encodes text and merge field values for use in JavaScript inside HTML tags by replacing characters that are reserved in HTML with HTML entity equivalents and inserting escape characters before unsafe JavaScript characters. `JSINHTMLENCODE(`*`someValue`*`)` is a convenience function that is equivalent to `JSENCODE(HTMLENCODE((`*`someValue`*`))`. That is, `JSINHTMLENCODE` first encodes *someValue* with `HTMLENCODE`, and then encodes the result with `JSENCODE`.
URLENCODE	Encodes text and merge field values for use in URLs by replacing characters that are illegal in URLs, such as blank spaces, with the code that represent those characters as defined in *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax*. For example, blank spaces are replaced with `%20`, and exclamation points are replaced with `%21`.