Finance-Based Functions

Introduction

Microsoft Access provides a series of function destined to perform various types of financially related operations. These functions use common factors depending on the value that is being calculated. Many of these functions deal with investments or loan financing.

The **Present Value** is the current value of an investment or a loan. For a savings account, a customer could pledge to make a set amount of deposit on a bank account every month. The initial value that the customer deposits or has in the account is the **Present Value**. The sign of the variable, when passed to a function, depends on the position of the customer. If the customer is making deposits, this value must be negative. If the customer is receiving money (lottery installment, family inheritance, etc), this value should be positive.

The **Future Value** is the value the loan or investment will have when the loan is paid off or when the investment is over. For a car loan, a musical instrument loan, a financed refrigerator, a boat, etc, this is usually 0 because the company that is lending the money will not take that item back (they didn't give it to the customer in the first place, they only lend him or her some money to buy the item). This means that at the end of the loan, the item (such as a car, boat, guitar, etc) belongs to the customer and it is most likely still worth something.

As describe above and in reality, the **Future Value** is the amount the item would be worth at the end. In most if not all loans, it would be 0. On the other hand, if a customer is borrowing money to buy something like a car, a boat, a piano, etc, the salesperson would ask if the customer wants to put a "down payment", which is an advance of money. Then, the salesperson or loan officer can either use that down payment as the **Future Value** parameter or simply subtract it from the **Present Value** and then apply the calculation to the difference. Therefore, you can apply some type of down payment to your functions to as the **Future Value**.

The Number Of **Periods** is the number of payments that make up a full cycle of a loan or an investment.

The Interest **Rate** argument is a fixed percent value applied during the life of the loan or the investment. The rate does not change during the length of the **Periods**.

It is very important to understand how these two arguments are passed to a function. The period could be the number of months of a year, which is 12; but it could be another length. Suppose a customer is getting a car loan that would be financed in 5 years. This is equivalent to 5 * 12 = 60 months. In the same way, a cash loan can stretch from 0 to 18 months, a carpenter truck loan can have a life financing of 40 months, and a condominium can be financed for 15 years of 12 months plus an additional 8 months; this is equivalent to (15 * 12) + 8 = 188 months. Here is the tricky part, especially as far as Microsoft Excel deals with its finance functions. If you pass the number of **Periods** in terms of years, such as 5 for a car loan that stretches over 5 years, then you can pass the **Rate** as a percentage value, such as 8.75%. If you pass the number of Periods in terms of months, for example you can pass it as 44 for a car that is financed in 3 years and 8 months, then you must communicate this to the Rate argument by dividing the Rate by 12. In other words, a Rate of 8.75% would be passed as 8.75%/12. If the Rate was typed in a cell named B2 that displays 8.75%, you can pass it as B2/12.

For deposits made in a savings account, because their payments are made monthly, the rate is divided by the number of periods (the **Periods**) of a year, which is 12. If an investment has an interest rate set at 14.50%, the **Rate** would be 14.50/12 = 1.208. Because the **Rate** is a percentage value, its actual value must be divided by 100 before passing it to the function. For a loan of 14.50% interest rate, this would be 14.50/12 = 1.208/100 = 0.012.

The **Payment** is the amount the customer will be paying. For a savings account where a customer has pledged to pay a certain amount in order to save a set (goal) amount, this would be the amount the customer would pay every month. If the customer is making payments (car loan, mortgage, deposits to a savings account, etc), this value must be negative. If the customer is receiving money (lottery installment or annuity, family inheritance, etc), this value must be positive.

The Payment **Time** specifies whether the payment is made at the beginning or the end of the period. For a monthly payment, this could be the beginning or end of every month.

The Future Value of an Investment

To calculate the future value of an investment, you can use the **FV()** function. The syntax of this function is:

```
FV(Rate, Periods, Payment, PresentValue, PaymentType)
```

The Number of Periods of an Investment

To calculate the number of periods of an investment or a loan, you can use the **NPer()** function. Its syntax is:

```
NPer(Rate, Payment, PresentValue, FutureValue, PaymentType);
```

Investment or Loan Payment

The **Pmt()** function is used to calculate the regular payment of loan or an investment. Its syntax is:

```
Pmt(Rate, NPeriods, PresentValue, FutureValue, PaymentType)
```

In the following example, a customer is applying for a car loan. The cost of the car will be entered in cell C4. It will be financed at a rate entered in cell C6 for a period set in cell C7. The dealer estimates that the car will have a value of $0.00 when it is paid off.

The Amount Paid As Interest During a Period

When a customer is applying for a loan, an investment company must be very interested to know how much money it would collect as interest. This allows the company to know whether the loan is worth giving. Because the interest earned is related to the interest rate, a company can play with the rate (and also the length) of the loan to get a fair (?) amount.

The **IPmt()** function is used to calculate the amount paid as interest on a loan during a period of the lifetime of a loan or an investment. It is important to understand what this function calculates. Suppose a customer is applying for a car loan and the salesperson decides (or agrees with the customer) that the loan will be spread over 5 years (5 years * 12 months each = 60 months). The salesperson then applies a certain interest rate. The **IPMT()** function can help you calculate the amount of interest that the lending institution would earn during a certain period. For example, you can use it to know how much money the company would earn in the 3rd year, or in the 4th year, or int the 1st year. Based on this, this function has an argument called Period, which specifies the year you want to find out the interest earned in.

The syntax of the **IPmt()** function is:

```
IPmt(Rate, Period, NPeriods, PresentValue, FutureValue, PaymentType)
```

The *Rate* argument is a fixed percent value applied during the life of the loan.

The *PresentValue* is the current value of the loan or investment. It could be the marked value of the car, the current mortgage value of a house, or the cash amount that a bank is lending.

The *FutureValue* is the value the loan or investment will have when the loan is paid off.

The *NPeriods* is the number of periods that occur during the lifetime of the loan. For example, if a car is financed in 5 years, this value would be (5 years * 12 months each =) 60 months. When passing this argument, you must remember to pass the right amount.

The *Period* argument represents the payment period. For example, it could be 3 to represent the 3rd year of a 5 year loan. In this case, the **IPmt()** function would calculate the interest earned in the 3rd year only.

The *PaymentType* specifies whether the periodic (such as monthly) payment of the loan is made at the beginning (1) or at the end (1) of the period.

The *FutureValue* and the *PaymentType* arguments are not required.

The Amount Paid as Principal

While the **IPmt()** function calculates the amount paid as interest for a period of a loan or an investment, the **PPmt()** function calculates the actual amount that applies to the balance of the loan. This is referred to as the principal. Its syntax is:

```
PPMT(Rate, Period, NPeriods, PresentValue, FutureValue, PaymentType)
```

The argument are the same as described in the previous sections.

The Present Value of a Loan or an Investment

The **PV()** function calculates the total amount that future investments are worth currently. Its

syntax is:

```
PV(Rate, NPeriods, Payment, FutureValue, PaymentType)
```

The arguments are the same as described earlier.

The Interest Rate

Suppose a customer comes to your car dealer and wants to buy a car. The salesperson would first present the available cars to the customer so the customer can decide what car he likes. After this process and during the evaluation, the sales person may tell the customer that the monthly payments would be $384.48. The customer may then say, "Wooooh, I can't afford that, man". Then the salesperson would ask, "What type of monthly payment suits you". From now on, both would continue the discussion. Since the salesperson still wants to make some money but without losing the customer because of a high monthly payment, the salesperson would need to find a reasonable rate that can accommodate an affordable monthly payment for the customer.

The **Rate()** function is used to calculate the interest applied on a loan or an investment. Its syntax is:

```
RateE(NPeriods, Payment, PresentValue, FutureValue, PaymentType, Guess)
```

All of the arguments are the same as described for the other functions, except for the *Guess*. This argument allows you to give some type of guess for a rate. This argument is not required. If you omit it, its value is assumed to be 10.

The Internal Rate of Return

The **IRR()** function is used to calculate an internal rate of return based on a series of investments. Its syntax is:

```
IRR(Values, Guess)
```

The *Values* argument is a series (also called an array or a collection) of cash amounts that a customer has made on an investment. For example, a customer could make monthly deposits in a savings or credit union account. Another customer could be running a business and receiving different amounts of money as the business is flowing (or losing money). The cash flows don't have to be the same at different intervals but they should (or must) occur at regular intervals such as weekly (amount cut from a paycheck), bi-weekly (401k directly cut from paycheck, monthly (regular investment), or yearly (income). The *Values* argument must be passed as a collection of values, such as a range of selected cells, and not an amount. Otherwise you would receive an error.

The *Guess* parameter is an estimate interest rate of return of the investment.

The Net Present Value

The **NPV()** function uses a series of cash flows to calculate the present value of an investment. Its syntax is:

```
NPV(Rate, Value1, Value2, ...)
```

The *Rate* parameter is the rate of discount in during one period of the investment.

As the NPV() function doesn't take a fixed number of arguments, you can add a series of values as Value1, Value2, etc. These are regularly made payments for each period involved. Because this function uses a series of payments, any payment made in the past should have a positive value (because it was made already). Any future payment should have a negative value (because it has not been made yet).

MOUS Topics

S17	Use the Control Toolbox to add controls
S31	Create a calculated field

Exercises

Watts A loan

1. Open the Watts A Loan database.
   Open the CustomersTransactions form in Design View. Add a Text Box below the subform. Set its Name to **txtTotalPayments** then set its **Format** to **Currency** with 2 **Decimal Places**. Make it get its value from the **txtTransactions** text box of the **sbfAccountTransactions** form
   Add another Text Box below the subform and change its properties as follows:
   Name: txtCurrentBalance

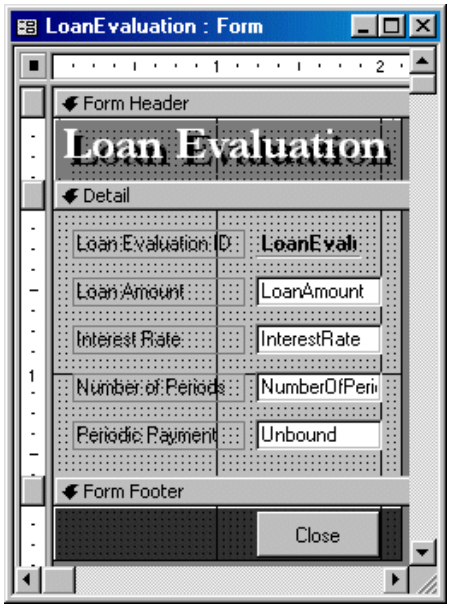
Control Source: =DLookUp("LoanAmount", "LoanProcessing", "CustomerID = " & CustomerID) - Nz(txtTotalPayments)
Format: Currency
Decimal Places: 2
Save and close the form

2. Create a new table in Design View with the following fields:

Field Name	Data Type	Additional Information
LoanEvaluationID	AutoNumber	Primary Key Caption: Loan Evaluation ID
LoanAmount	Currency	Caption: Loan Amount Default Value: 0
InterestRate	Number	Field Size: Double Format: Percent Caption: Interest Rate Default Value: 0.0875
NumberOfPeriods	Number	Field Size: Integer Caption: NumberOfPeriods Default Value: 12

Save the table as LoanEvaluation and close it

3. Create a new form based on the LoanEvaluation table
Save the form as **LoanEvaluation**
Add a Text Box in its Detail section and set its properties as follows:
Name: txtPeriodicPayment
Control Source:
=Abs(Pmt(Nz(CDbl([InterestRate]))/12, Nz(CInt([NumberOfPeriods])),Nz(CDbl([LoanAmount])),0,0))
Format: Currency
Decimal Places: 2
Use the Command Button Wizard to add a button that can be used to close the form
Design the form as follows:

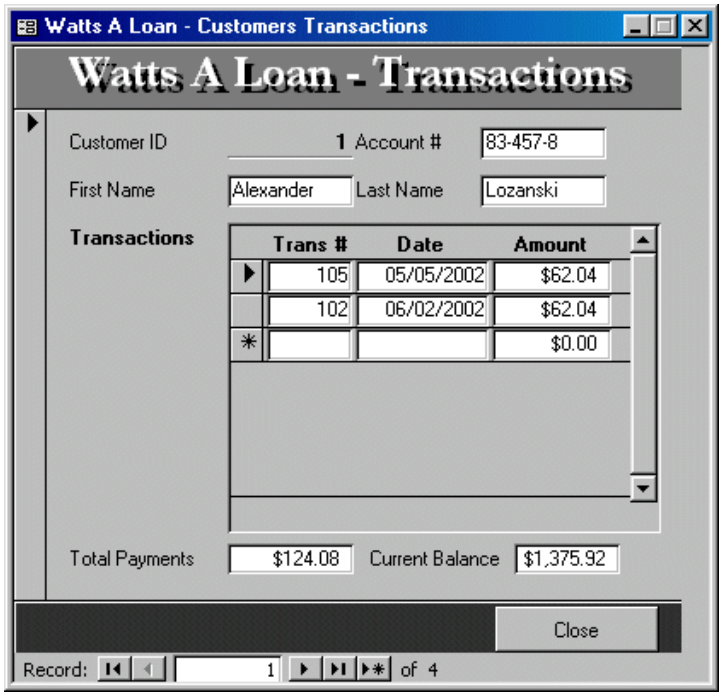


Disable the Maximize button and make the form Pop Up. Don't make it automatically center itself. Before saving the form, position it slightly to the middle-right side of the screen so Microsoft Access would remember that position
Save and close the form

4. Open the LoanProcessing form in Design View. Using the Command Button Wizard to add a button that, when clicked, would open the LoanEvaluation form (remember that there is no relationship between both forms; therefore, you will Open The Form And Show All The Records). Set the button's Text to Loan Evaluation and its Name to cmdLoanEval.
Save and close the form

5. Open the LoanProcessing form and use its Loan Evaluation button to open the LoanEvaluation form. In the Loan Evaluation form, evaluate a $1500.00 amount of at 12.50% paid in 28 months. After evaluating it, manually create a new personal loan in the Loan Processing form for the amount of $1500 at 12.50% for 28 payments. The loan is processed by the owner, for the 83-457-8 account on April 10th, 2002. Make the 1st payment due on May 20th of the same year and put a reminder that the payments are due every 22th of the month
Evaluate other amounts and create loans for the other customers.
Close both forms

6. Open the CustomersTransactions form to see the results



Previous