1 Credits

This tutorial Is based on:

- ADO Tutorial written by - **H Jeske.**
- ADO Lessons written by - **Gauteng Education Department**

2 Contents

3 Introduction

ADO is another way of interacting with a relational database through Delphi. For your exam preparation, you must know the following:

- How to insert/delete/update a table
- How to do calculations on values retrieved from the database

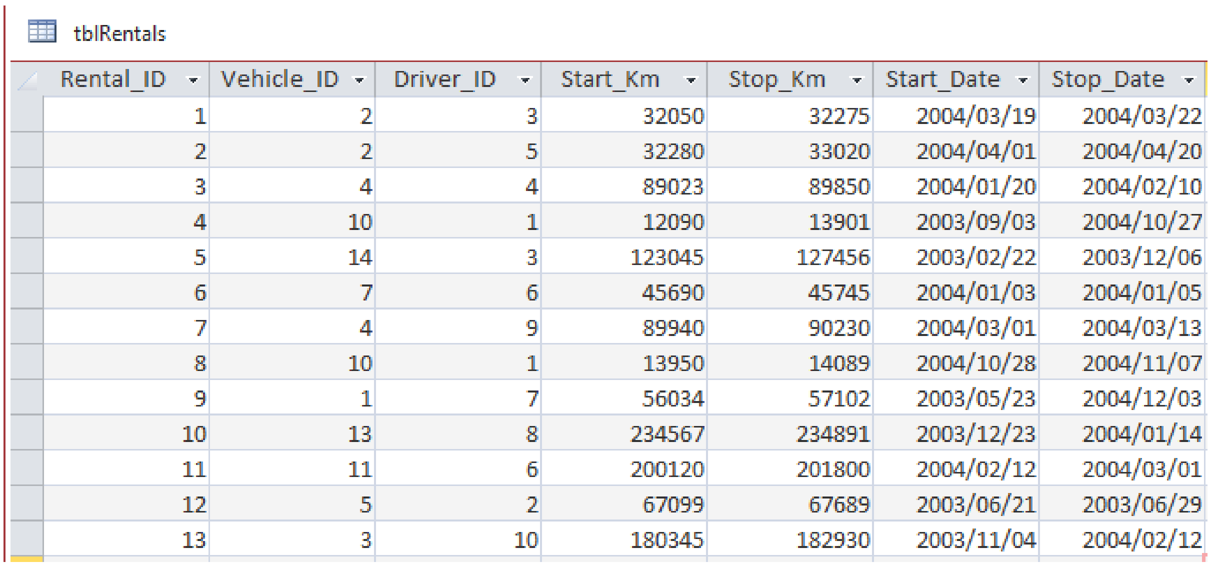
The exam questions will come connected to the database already, so you will not need to worry need to worry about that.

4 The Car Rental Database

The car rental database contains three tables:

- tblRentals
- tblVehicles
- tblRates

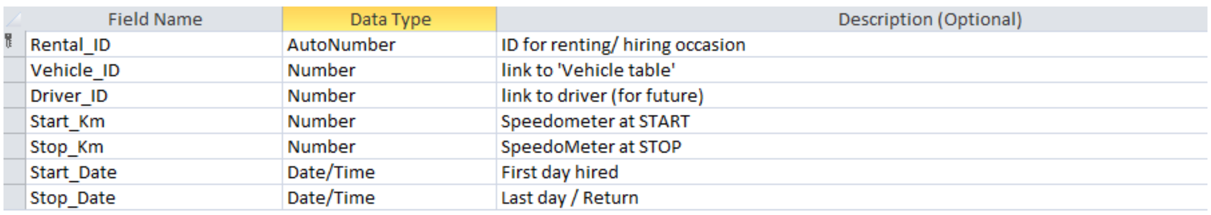
We will only be focussing on one table, namely *tblRentals* – to demonstrate how to use ADO.



tblRentals

Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
1	2	3	32050	32275	2004/03/19	2004/03/22
2	2	5	32280	33020	2004/04/01	2004/04/20
3	4	4	89023	89850	2004/01/20	2004/02/10
4	10	1	12090	13901	2003/09/03	2004/10/27
5	14	3	123045	127456	2003/02/22	2003/12/06
6	7	6	45690	45745	2004/01/03	2004/01/05
7	4	9	89940	90230	2004/03/01	2004/03/13
8	10	1	13950	14089	2004/10/28	2004/11/07
9	1	7	56034	57102	2003/05/23	2004/12/03
10	13	8	234567	234891	2003/12/23	2004/01/14
11	11	6	200120	201800	2004/02/12	2004/03/01
12	5	2	67099	67689	2003/06/21	2003/06/29
13	3	10	180345	182930	2003/11/04	2004/02/12

You will be given a similar table in your exam, take note of the columns, and their data types.

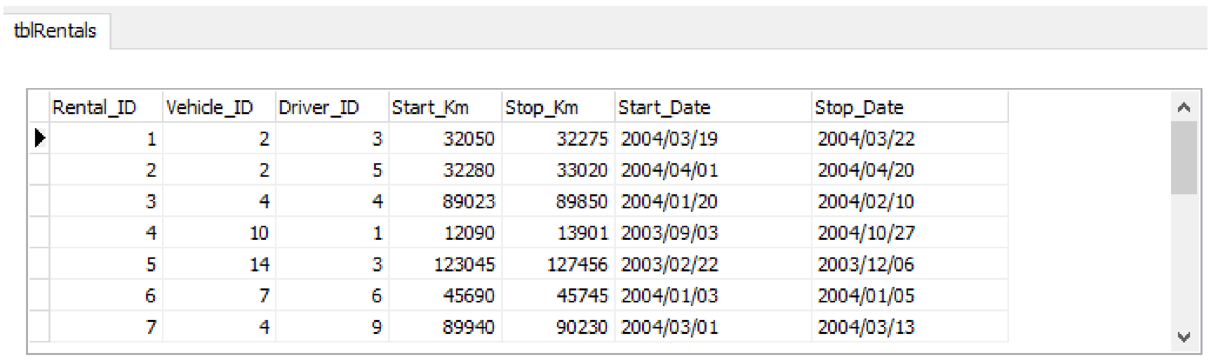


Field Name	Data Type	Description (Optional)
Rental_ID	AutoNumber	ID for renting/ hiring occasion
Vehicle_ID	Number	link to 'Vehicle table'
Driver_ID	Number	link to driver (for future)
Start_Km	Number	Speedometer at START
Stop_Km	Number	SpeedoMeter at STOP
Start_Date	Date/Time	First day hired
Stop_Date	Date/Time	Last day / Return

5 ACCESSING A SINGLE VALUE IN THE TABLE

The first thing we will do is learn how to access values in the database table.

The table will given to you in this format :


tblRentals

	Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
▶	1	2	3	32050	32275	2004/03/19	2004/03/22
	2	2	5	32280	33020	2004/04/01	2004/04/20
	3	4	4	89023	89850	2004/01/20	2004/02/10
	4	10	1	12090	13901	2003/09/03	2004/10/27
	5	14	3	123045	127456	2003/02/22	2003/12/06
	6	7	6	45690	45745	2004/01/03	2004/01/05
	7	4	9	89940	90230	2004/03/01	2004/03/13

Now notice the **little black arrow** on the left hand side of the table.

This arrow, is known as the **pointer** and shows the *current selected record*. In this case, the first record with *Rental_ID = 1* is selected.

Accessing a records value is done, using the following syntax :

```
tblRentals['Rental_ID']
```

If we wanted to do something with this value, like for example print it to a rich edit we must take note of the **data_type** of that column. In this case, *Rental_ID* is a number therefore we will have to convert it to a integer :

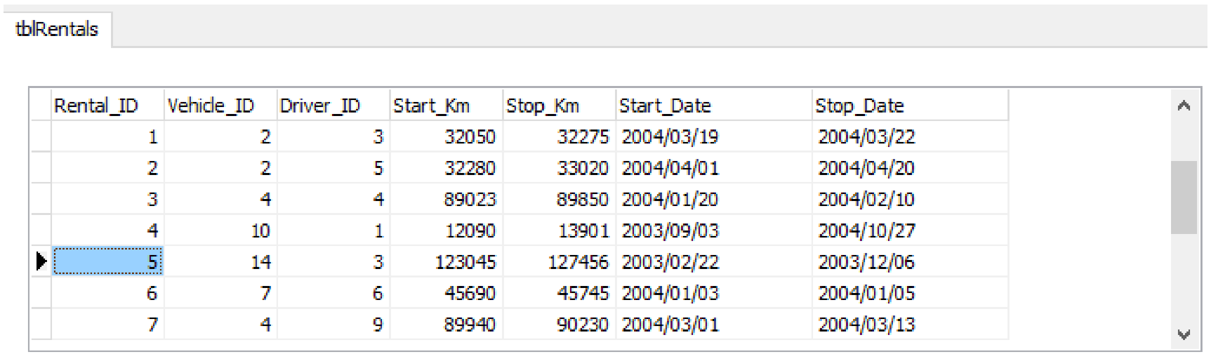
```
redOutput.Lines.Add('Rental_ID:' + #9 +  inttostr(tblRentals['Rental_ID']));
```

In the case of accessing a Date Column such as *Start_Date* we can use :

```
redOutput.Lines.Add('Start_Date:' + #9 + datetostr(tblRentals['Start_Date']));
```

NOTE: If you click on the table itself, and **select a record**, the black pointer will then point to that record you selected – and if you print to the rich edit, that will be the records values that will be printed.

Here we have clicked on a record, where *Rental_ID = 5* :



tblRentals

	Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
	1	2	3	32050	32275	2004/03/19	2004/03/22
	2	2	5	32280	33020	2004/04/01	2004/04/20
	3	4	4	89023	89850	2004/01/20	2004/02/10
	4	10	1	12090	13901	2003/09/03	2004/10/27
▶	5	14	3	123045	127456	2003/02/22	2003/12/06
	6	7	6	45690	45745	2004/01/03	2004/01/05
	7	4	9	89940	90230	2004/03/01	2004/03/13

And if you print the values for this record :

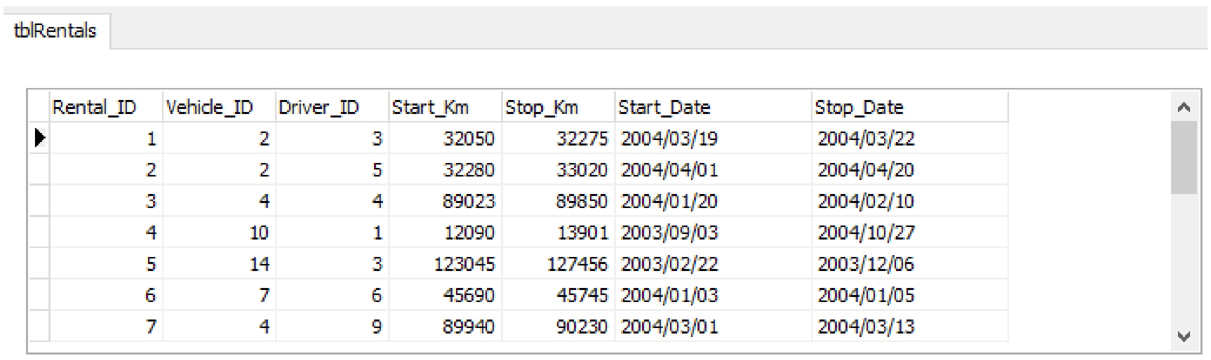
```
procedure TfrmCarRent.btnPrintClick(Sender: TObject);
begin
  redOutput.Lines.Clear;
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 80;

  redOutput.Lines.Add('Rental_ID:' + #9 +  inttostr(tblRentals['Rental_ID']));
  redOutput.Lines.Add('Vehicle_ID:' + #9 + inttostr(tblRentals['Vehicle_ID']));
  redOutput.Lines.Add('Driver_ID:' + #9 + inttostr(tblRentals['Driver_ID']));
  redOutput.Lines.Add('Start_Km:' + #9 +  inttostr(tblRentals['Start_Km']));
  redOutput.Lines.Add('Stop_Km:' + #9 + inttostr(tblRentals['Stop_Km']));
  redOutput.Lines.Add('Start_Date:' + #9 + datetostr(tblRentals['Start_Date']));
  redOutput.Lines.Add('Stop_Date:' + #9 + datetostr(tblRentals['Stop_Date']));

end;
```

Which will give the output :

```
Rental_ID:          5
Vehicle_ID:         14
Driver_ID:          3
Start_Km:           123045
Stop_Km:            127456
Start_Date:         2003/02/22
Stop_Date:          2003/12/06
```

6 ACCESSING ALL THE RECORDS IN THE TABLE

This is our table again :



tblRentals

Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
1	2	3	32050	32275	2004/03/19	2004/03/22
2	2	5	32280	33020	2004/04/01	2004/04/20
3	4	4	89023	89850	2004/01/20	2004/02/10
4	10	1	12090	13901	2003/09/03	2004/10/27
5	14	3	123045	127456	2003/02/22	2003/12/06
6	7	6	45690	45745	2004/01/03	2004/01/05
7	4	9	89940	90230	2004/03/01	2004/03/13

How would we work out the total amount of *Start_Km* in the table ?

If we had to do it with a calculator, we would:

1. Start at the top of the table
2. Add each Start_Km value
3. Until you reach the bottom of the table, we you have the total sum

ADO provides some **Navigation** commands, that allow us to move the pointer without having to click each time on the table :

```
    tblRentals.First          // Moves pointer to the first record
    tblRentals.Prior          // Moves pointer to record one before current record
    tblRentals.Next           // Moves pointer to the next record
    tblRentals.Last           // Moves pointer to the last record
```

To simply make use of these functions, we can just call them :

```
procedure TfrmCarRent.btnRentalsFirstClick(Sender: TObject);
begin
  tblRentals.First;
end;
```

That would set the pointer to the first record. Back to our question, how to work out the total *Start_Km's?*

- The **tblRentals.First** method will useful when we want to start at the top of the table.
- The **tblRentals.Next** method will be useful to move through all our records

But now we cant say tblRentals.Next for every record in the table, that's to tedious. Therefore we will use a *while* loop.

The *while* loop, takes on the following structure:

```
while not tblRentals.Eof do
begin

end;
```

The **Eof** is a *Boolean property* and so the while loop will execute until the *End of file* is reached. Currently, the while loop will result in a infinite loop – because it will read the tblRentals, but never reach the end of the file. This is where our ***Next*** and ***First*** *methods* come in.

```
  tblRentals.First;
  while not tblRentals.Eof do
  begin
       tblRentals.Next;
  end;
```

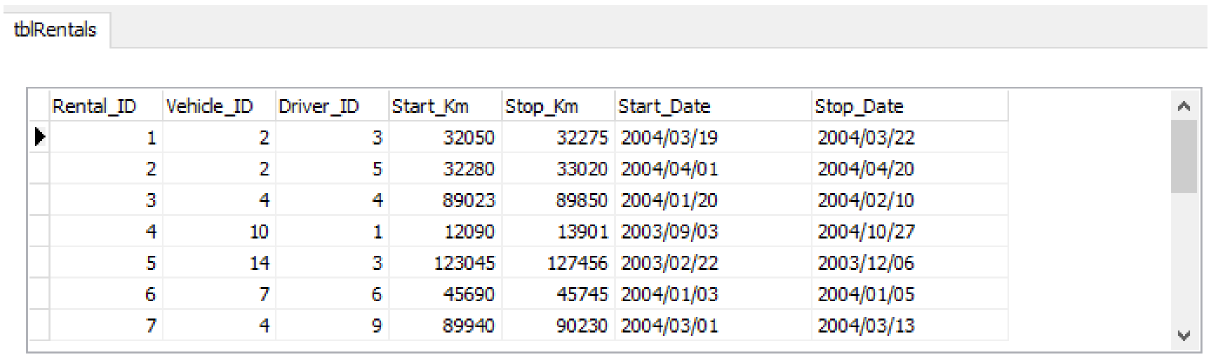
This will start at the top of the table (because of First) and proceed through each record in the table (because of Next).

```
procedure TfrmCarRent.btnTotalStartKmClick(Sender: TObject);
var iSum : integer;
begin
  iSum := 0;
  tblRentals.First;
  while not tblRentals.Eof do
  begin
    iSum := iSum  + tblRentals['Start_Km'];
    tblRentals.Next;
  end;

  redOutput.Clear;
  redOutput.Lines.Add('Total Start_Km = ' + inttostr(iSum) + ' Kms');

end;
```

Which will give the output :

```
Total Start_Km = 1176233 Kms
```

7 COUNTING THE TOTAL AMOUNT OF ENTRIES IN A TABLE


tblRentals

Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
1	2	3	32050	32275	2004/03/19	2004/03/22
2	2	5	32280	33020	2004/04/01	2004/04/20
3	4	4	89023	89850	2004/01/20	2004/02/10
4	10	1	12090	13901	2003/09/03	2004/10/27
5	14	3	123045	127456	2003/02/22	2003/12/06
6	7	6	45690	45745	2004/01/03	2004/01/05
7	4	9	89940	90230	2004/03/01	2004/03/13

We want to know how many records are present in the *tblRentals* table. Since we learnt about the usage of a *while* loop, in the last section we can use it!

```
procedure TfrmCarRent.btnTotalRecordsClick(Sender: TObject);
var iTotalRecords : integer;
begin
  iTotalRecords := 0;
  tblRentals.First;
  while not tblRentals.Eof do
  begin
    inc(iTotalRecords);
    tblRentals.Next;
  end;
  redOutput.Clear;
  redOutput.Lines.Add('TotalRecords = ' + inttostr(iTotalRecords));
end;
```

Which would output:

```
TotalRecords = 13
```

There is a built-in method that does the exact same thing called **RecordCount:**

```
procedure TfrmCarRent.btnRecordCountClick(Sender: TObject);
begin
  redOutput.Clear;
  redOutput.Lines.Add('TotalRecords = ' +
inttostr(tblRentals.RecordCount));
end;
```

8 MODIFY COMMANDS

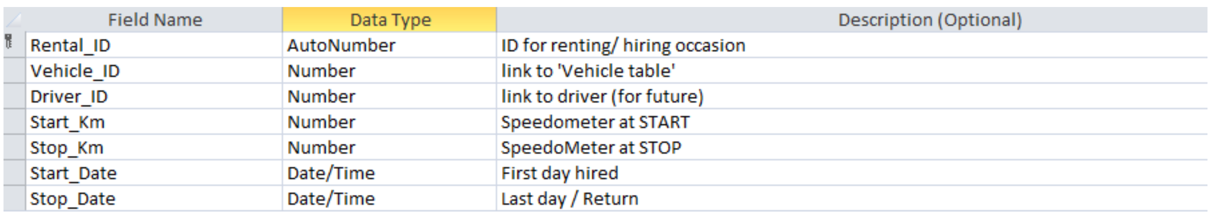
The modify commands, correspond to the SQL Update, Insert and Delete. Here the ones you will need to learn for ADO:

```
tblPlayers.Edit       // Puts the dataset in edit mode
tblPlayers.Insert     // Puts the dataset in insert mode
tblPlayers.Delete     // Puts the dataset in delete mode
tblPlayers.Post       // Writes the change to current record to the database
```

Edit is to Update a table, and Post is to save the changes. We are going to look at some code examples, of how to implement these commands.

9 INSERTING A RECORD INTO THE TABLE

If you can recall, at the start of this tutorial we spoke about data types when introducing the table *tblRentals.*



Field Name	Data Type	Description (Optional)
Rental_ID	AutoNumber	ID for renting/ hiring occasion
Vehicle_ID	Number	link to 'Vehicle table'
Driver_ID	Number	link to driver (for future)
Start_Km	Number	Speedometer at START
Stop_Km	Number	SpeedoMeter at STOP
Start_Date	Date/Time	First day hired
Stop_Date	Date/Time	Last day / Return

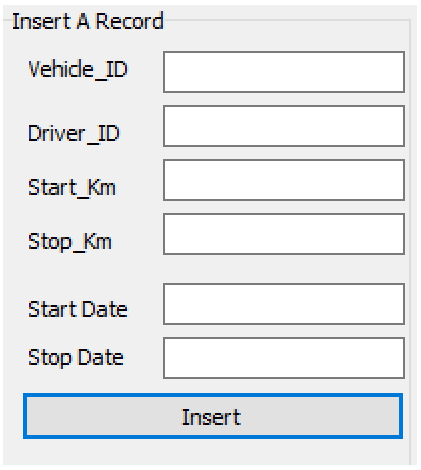
For an **autonumber** column, you do no need to insert anything – access will automatically insert the value for you – but for the rest of the columns we will have to put something in.

The basic format for a insert statement is :

```
procedure TfrmCarRent.btnInsertClick(Sender: TObject);
begin
  tblRentals.Insert;
  tblRentals.Post;
end;
```

The **Insert** statement is responsible for putting the table into insert mode, and **Post** statement is responsible for saving our any changes. If we run this code, we will end up with an error because there are no values!

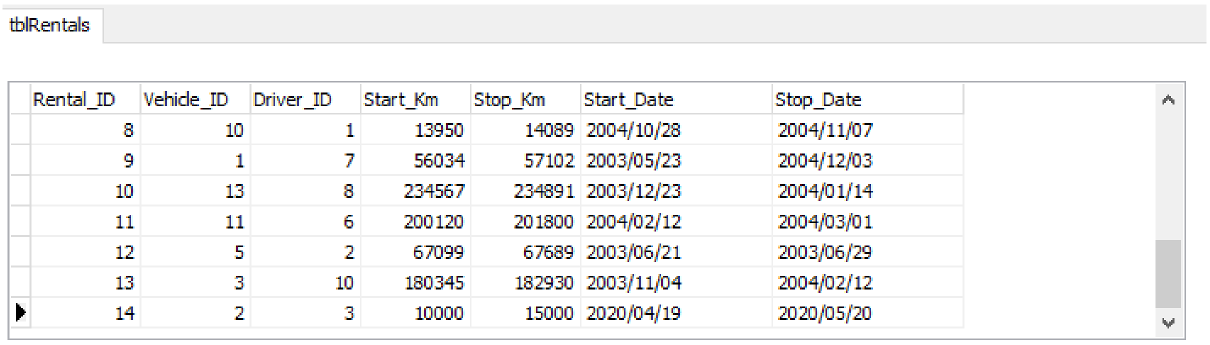
To help us enter in some values, there is this nice gui here :



When we click on the **Insert** button, the values from the Editboxes will be insert into the table.

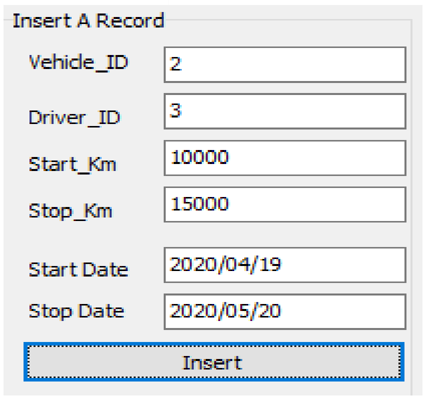
```
procedure TfrmCarRent.btnInsertClick(Sender: TObject);
begin
  tblRentals.Insert;
  tblRentals['Vehicle_ID'] := strtoint(edtVehicle_ID.Text);
  tblRentals['Driver_ID'] := strtoint(edtDriver_ID.Text);
  tblRentals['Start_Km'] := strtoint(edtStart_Km.Text);
  tblRentals['Stop_Km'] := strtoint(edtStop_Km.Text);
  tblRentals['Start_Date'] := strToDate(edtStart_Date.Text);
  tblRentals['Stop_Date'] := strToDate(edtStop_Date.Text);
  tblRentals.Post;
end;
```

This would input a new record into the table, as seen here by *Rental_ID = 14*:


tblRentals

Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
8	10	1	13950	14089	2004/10/28	2004/11/07
9	1	7	56034	57102	2003/05/23	2004/12/03
10	13	8	234567	234891	2003/12/23	2004/01/14
11	11	6	200120	201800	2004/02/12	2004/03/01
12	5	2	67099	67689	2003/06/21	2003/06/29
13	3	10	180345	182930	2003/11/04	2004/02/12
14	2	3	10000	15000	2020/04/19	2020/05/20

And our GUI values, we entered in like this :



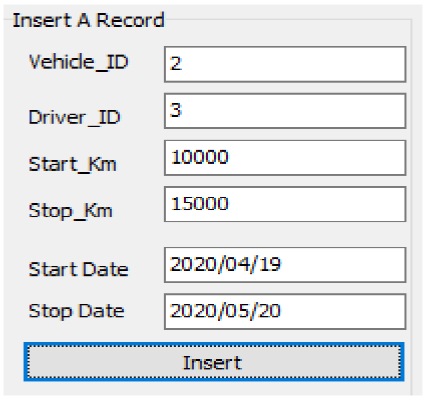
Insert A Record

Vehicle_ID	2
Driver_ID	3
Start_Km	10000
Stop_Km	15000
Start Date	2020/04/19
Stop Date	2020/05/20

10 Inserting a Record into the Table Manually

In the previous example, we used a EditBoxes, and converter functions to insert a new record into the table. However, in this section we are going to look at inserting a record into the table manually, or rephrased hard coded.

These are the values we had used in the previous example:



Vehicle_ID	2
Driver_ID	3
Start_Km	10000
Stop_Km	15000
Start Date	2020/04/19
Stop Date	2020/05/20

Hard coding these values would look like this :

```
procedure TfrmCarRent.btnInsertHardCodeClick(Sender: TObject);
begin

  tblRentals.Insert;
  tblRentals['Vehicle_ID'] := 2;
  tblRentals['Driver_ID'] := 3;
  tblRentals['Start_Km'] := 10000;
  tblRentals['Stop_Km'] := 15000;
  tblRentals['Start_Date'] := '2020/04/19';
  tblRentals['Stop_Date'] := '2020/05/20';
  tblRentals.Post;

  redOutput.Clear;
  redOutput.Lines.Add('New Record Succesfully Inserted');
end;
```

11 Edit Records in the Table

To edit a record in the table, we need to use two methods:

- **Edit:** Puts the table into edit mode (equivalent of *update* in SQL)
- **Post:** Saves any changes to the database

The corresponding code :

```
procedure TfrmCarRent.btnEditClick(Sender: TObject);
begin
  tblRentals.Edit;
  tblRentals.Post;
end;
```

We will take values from the GUI, and update a Record According to those values :

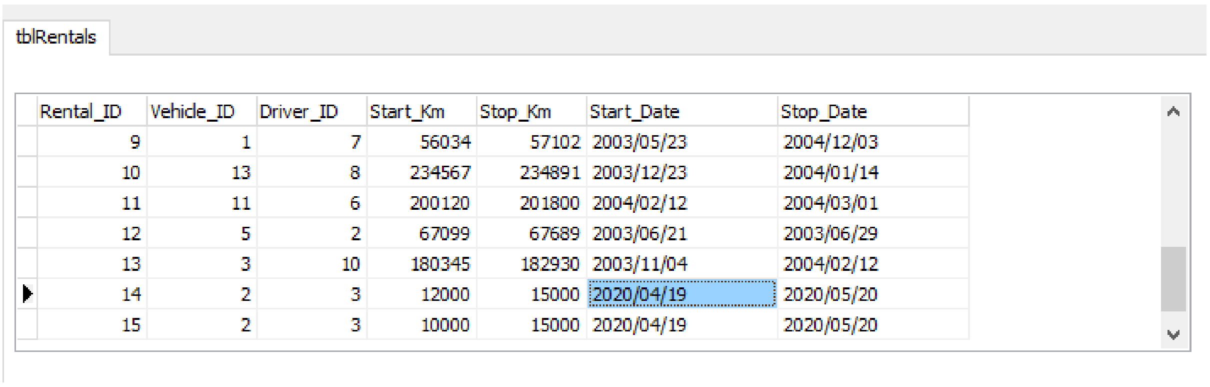
```
procedure TfrmCarRent.btnEditClick(Sender: TObject);
begin
  tblRentals.Edit;
    tblRentals['Vehicle_ID'] := strtoint(edtVehicle_ID.Text);
    tblRentals['Driver_ID'] := strtoint(edtDriver_ID.Text);
    tblRentals['Start_Km'] := strtoint(edtStart_Km.Text);
    tblRentals['Stop_Km'] := strtoint(edtStop_Km.Text);
    tblRentals['Start_Date'] := strToDate(edtStart_Date.Text);
    tblRentals['Stop_Date'] := strToDate(edtStop_Date.Text);
  tblRentals.Post;

  redOutput.Clear;
  redOutput.Lines.Add('New Record Succesfully Edited');
end;
```

12 Delete a Record from the table

To delete a record, we use the following format :

```
procedure TfrmCarRent.btnDeleteClick(Sender: TObject);
begin
  tblRentals.Delete;
end;
```

But often, the question will require us to delete based on a certain condition. Let us look at an example – say we wanted to delete all the records with the *Start_Date = 2020*



tblRentals

Rental_ID	Vehicle_ID	Driver_ID	Start_Km	Stop_Km	Start_Date	Stop_Date
9	1	7	56034	57102	2003/05/23	2004/12/03
10	13	8	234567	234891	2003/12/23	2004/01/14
11	11	6	200120	201800	2004/02/12	2004/03/01
12	5	2	67099	67689	2003/06/21	2003/06/29
13	3	10	180345	182930	2003/11/04	2004/02/12
14	2	3	12000	15000	2020/04/19	2020/05/20
15	2	3	10000	15000	2020/04/19	2020/05/20

First we will need a **While Loop**, to process every record in the table :

```
  tblRentals.First;
  while not tblRentals.Eof do
  begin
      tblRentals.Next;
  end;
```

The next question that arises, is how do we isolate those records with the year 2020 in the Start Date. If you recall, we can access the *Start_Date* field like this :

```
tblRentals['Start_Date']
```

And we can convert the value from the *Start_Date* column into a string like this :

```
datetostr(tblRentals['Start_Date']);
```

Once we have the string value from the *Start_Date* column, we can use the String function **Pos** to determine if the string contains the value 2020.

If the string is found, Pos will return a positive integer, otherwise it will return 0 if it finds nothing.

```
if pos('2020',datetostr(tblRentals['Start_Date'])) > 0 then
  begin

  end;
```

Our full code will now look this :

```
procedure TfrmCarRent.btnDeleteYearTwentyClick(Sender: TObject);
var iCount : integer;
begin
  iCount := 0;
  tblRentals.First;
  while not tblRentals.Eof do
  begin
      if pos('2020',datetostr(tblRentals['Start_Date'])) > 0 then
      begin
          tblRentals.Delete;
          inc(iCount);
          tblRentals.First;
      end;
      tblRentals.Next;
  end;
  redOutput.Lines.Add(inttostr(iCount) + ' Records Deleted');
end;
```