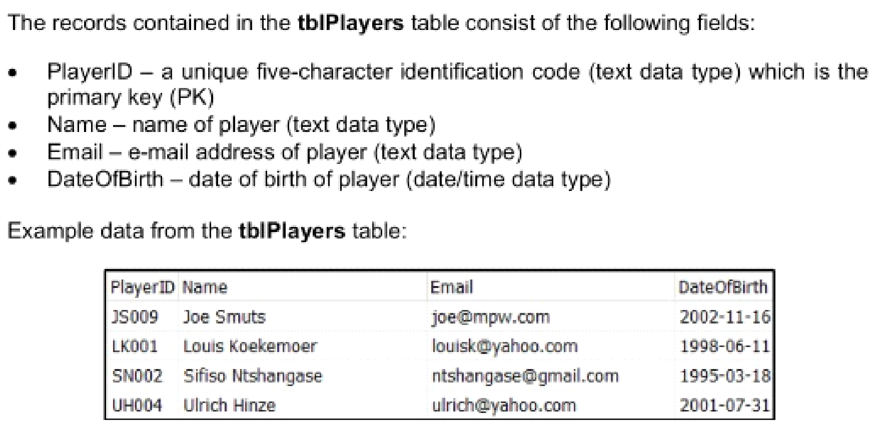
LAN_SQL

Thursday, 30 July 2020 01:29

Introduction

The LanFans.mdb (Database File) is password protected, therefore we will not be able to open it and examine its contents. We will therefore refer to the exam paper only, for information relating to the tables.

tblPlayers



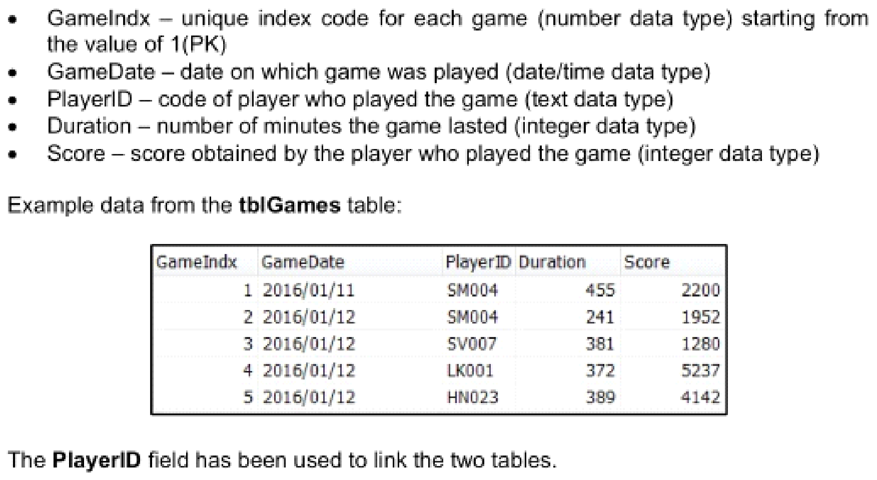
The records contained in the **tblPlayers** table consist of the following fields:

- PlayerID – a unique five-character identification code (text data type) which is the primary key (PK)
- Name – name of player (text data type)
- Email – e-mail address of player (text data type)
- DateOfBirth – date of birth of player (date/time data type)

Example data from the **tblPlayers** table:

PlayerID	Name	Email	DateOfBirth
JS009	Joe Smuts	joe@mpw.com	2002-11-16
LK001	Louis Koekemoer	louisk@yahoo.com	1998-06-11
SN002	Sifiso Ntshangase	ntshangase@gmail.com	1995-03-18
UH004	Ulrich Hinze	ulrich@yahoo.com	2001-07-31

tblGames



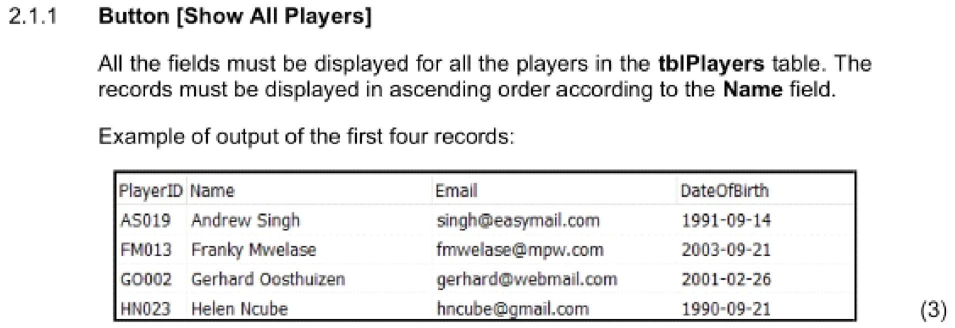
- GameIndx – unique index code for each game (number data type) starting from the value of 1(PK)
- GameDate – date on which game was played (date/time data type)
- PlayerID – code of player who played the game (text data type)
- Duration – number of minutes the game lasted (integer data type)
- Score – score obtained by the player who played the game (integer data type)

Example data from the **tblGames** table:

GameIndx	GameDate	PlayerID	Duration	Score
1	2016/01/11	SM004	455	2200
2	2016/01/12	SM004	241	1952
3	2016/01/12	SV007	381	1280
4	2016/01/12	LK001	372	5237
5	2016/01/12	HN023	389	4142

The **PlayerID** field has been used to link the two tables.

Question 2.1.1



2.1.1 **Button [Show All Players]**

All the fields must be displayed for all the players in the **tblPlayers** table. The records must be displayed in ascending order according to the **Name** field.

Example of output of the first four records:

PlayerID	Name	Email	DateOfBirth
AS019	Andrew Singh	singh@easymail.com	1991-09-14
FM013	Franky Mwelase	fmwelase@mpw.com	2003-09-21
GO002	Gerhard Oosthuizen	gerhard@webmail.com	2001-02-26
HN023	Helen Ncube	hncube@gmail.com	1990-09-21

(3)

Here will use a simple SELECT statement to get all the columns from tblPlayers, followed by a ORDER BY clause to sort by Name.

//

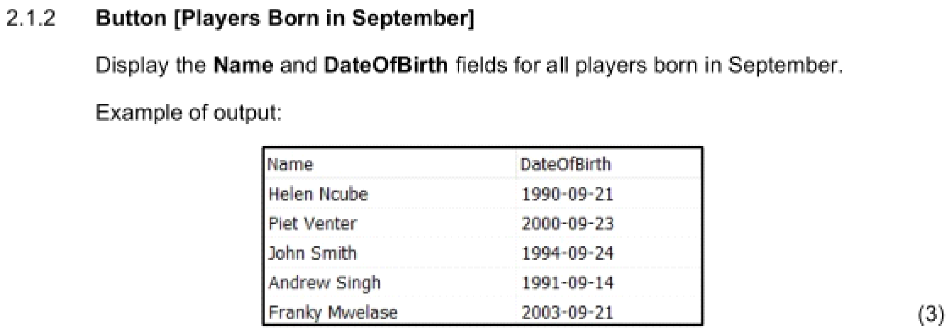
```
==========================================================================
====
//   QUESTION 2.1 - SQL
//
==========================================================================
====

procedure TfrmQuestion2.btnSQLqst211Click(Sender: TObject);

begin
   // Question 2.1.1
   sSQL := 'select * from tblPlayers order by Name;';

   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```

Question 2.1.2



2.1.2 **Button [Players Born in September]**

Display the **Name** and **DateOfBirth** fields for all players born in September.

Example of output:

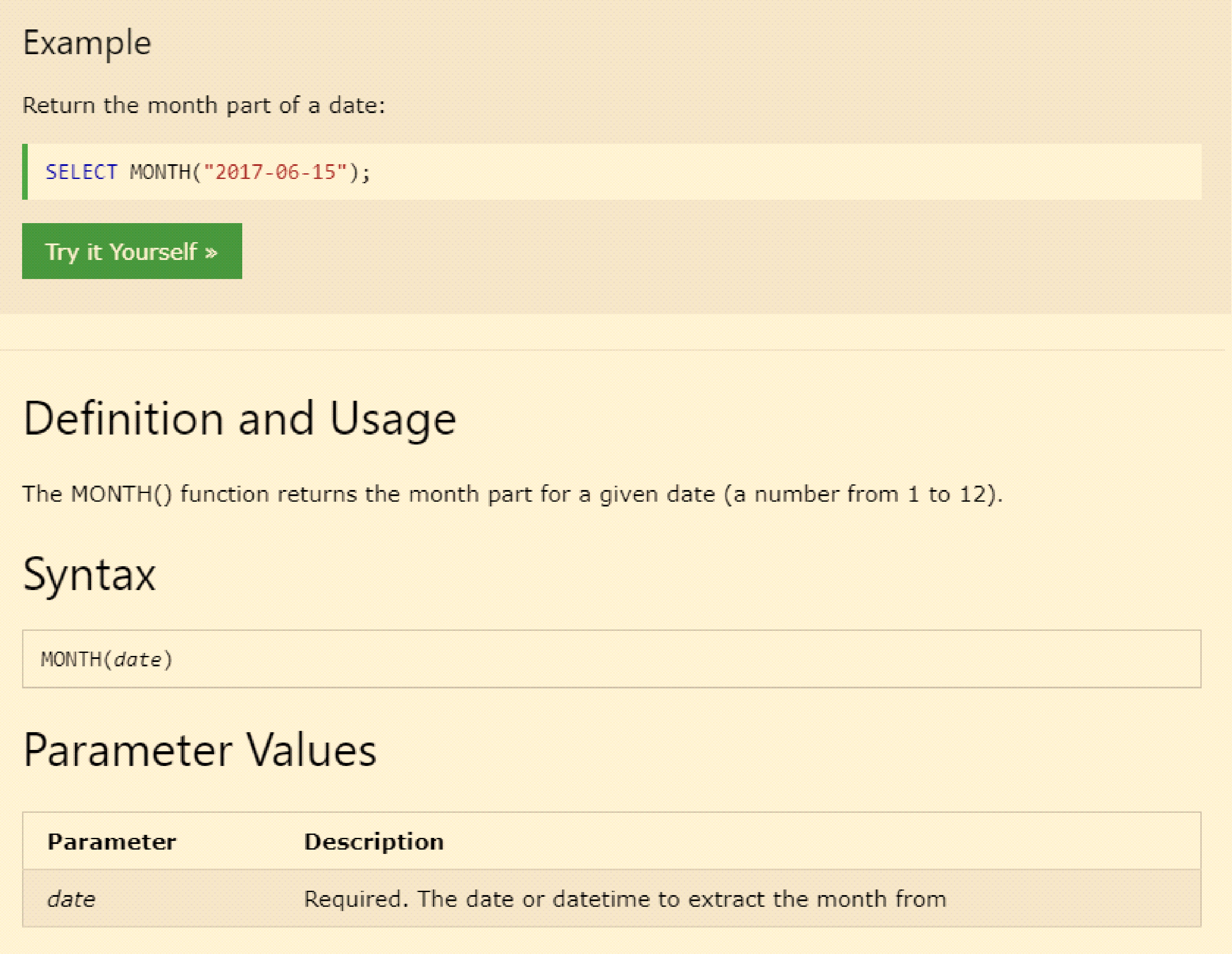
Name	DateOfBirth
Helen Ncube	1990-09-21
Piet Venter	2000-09-23
John Smith	1994-09-24
Andrew Singh	1991-09-14
Franky Mwelase	2003-09-21

(3)

To get the Name and DateOfBirth Columns, we can use a SELECT statement.

```
SELECT Name, DateOfBirth from tblPlayers;
```

However, the question requires DateOfBirth = September. SQL provides a Month Function to isolate the month :



Example

Return the month part of a date:

```
SELECT MONTH("2017-06-15");
```

Definition and Usage

The MONTH() function returns the month part for a given date (a number from 1 to 12).

Syntax

```
MONTH(date)
```

Parameter Values

Parameter	Description
date	Required. The date or datetime to extract the month from

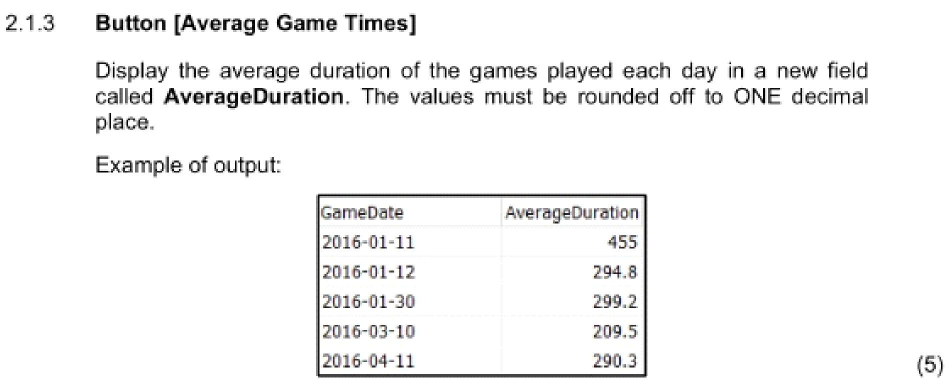
See : https://www.w3schools.com/sql/func_mysql_month.asp

Our SQL statement then changes to :

```
procedure TfrmQuestion2.btnSQLqst212Click(Sender: TObject);
begin
   // Question 2.1.2
   sSQL := 'select Name,DateOfBirth from tblPlayers where
Month(DateOfBirth)=9';

   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```

Question 2.1.3



2.1.3 **Button [Average Game Times]**

Display the average duration of the games played each day in a new field called **AverageDuration**. The values must be rounded off to ONE decimal place.

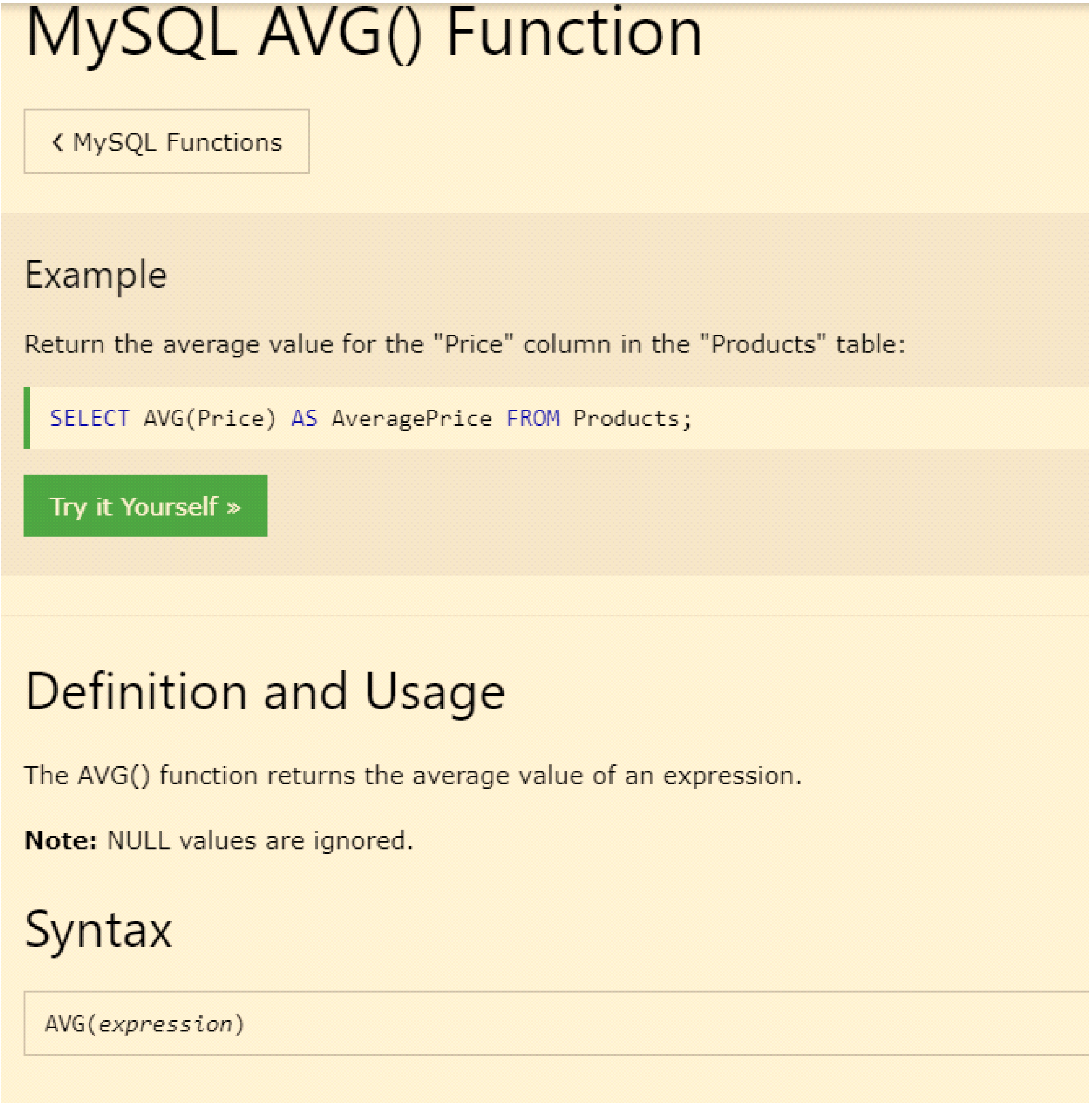
Example of output:

GameDate	AverageDuration
2016-01-11	455
2016-01-12	294.8
2016-01-30	299.2
2016-03-10	209.5
2016-04-11	290.3

(5)

Since there is no AverageDuration Column listed in tblGames, this a column we will have to create. We do have a Duration column, which has values for us to work with. Our task then , is to work out the Average Duration Per GameDate.

The average function will help use calculate the Average for the Duration column :



MySQL AVG() Function

Example

Return the average value for the "Price" column in the "Products" table:

```
SELECT AVG(Price) AS AveragePrice FROM Products;
```

Definition and Usage

The AVG() function returns the average value of an expression.

Note: NULL values are ignored.

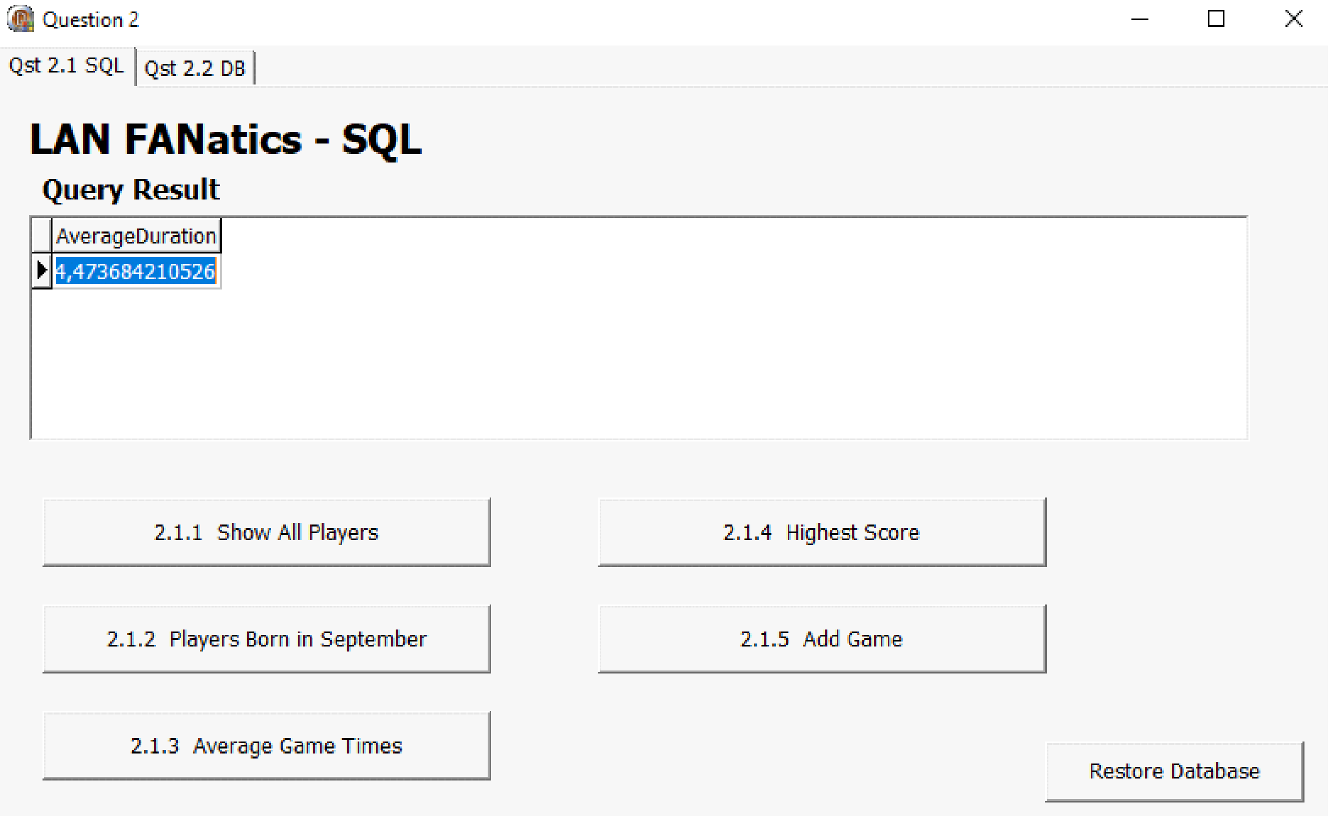
Syntax

```
AVG(expression)
```

See https://www.w3schools.com/sql/func_mysql_avg.asp

We can apply this definition above to our code, by stating :

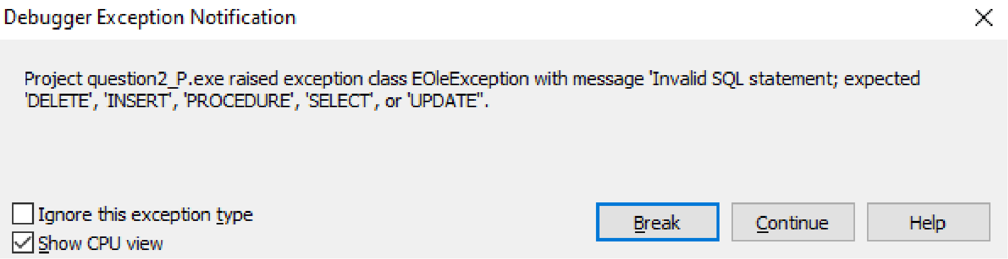
```
sSQL :=('select avg(duration) as AverageDuration from tblGames;');
```



AverageDuration
4,473684210526

What has happened here is our code has worked out the TOTAL AVERAGE for the Duration column. We need to work, AVG(Duration) per day. Lets try update our query, to include the GameDate column:

```
sSQL :=('select GameDate,avg(duration) as AverageDuration from
tblGames;');
```

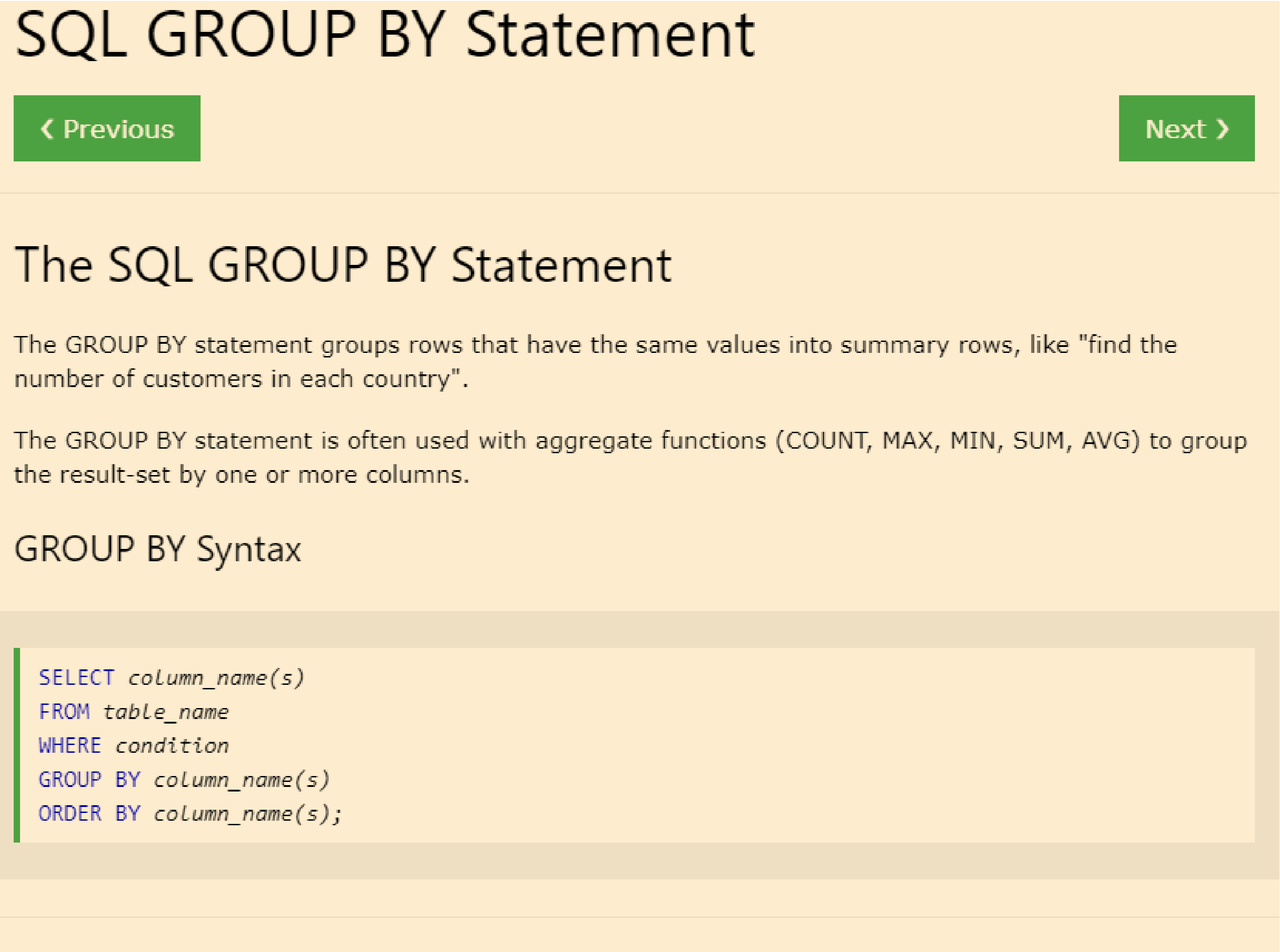


Project question2_P.exe raised exception class EOleException with message 'Invalid SQL statement; expected 'DELETE', 'INSERT', 'PROCEDURE', 'SELECT', or 'UPDATE''.

□ Ignore this exception type
☑ Show CPU view

Unfortunately, this gives us an error!. The problem here, is that we are using an AGGREGATE function AVG, and when we use an AGGREGATE function, we have to use the GROUP BY clause also in conjunction with the AGGREGATE function. Other aggregate functions include :

```
1) Count()
2) Sum()
3) Avg()
4) Min()
5) Max()
```



SQL GROUP BY Statement

The SQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

GROUP BY Syntax

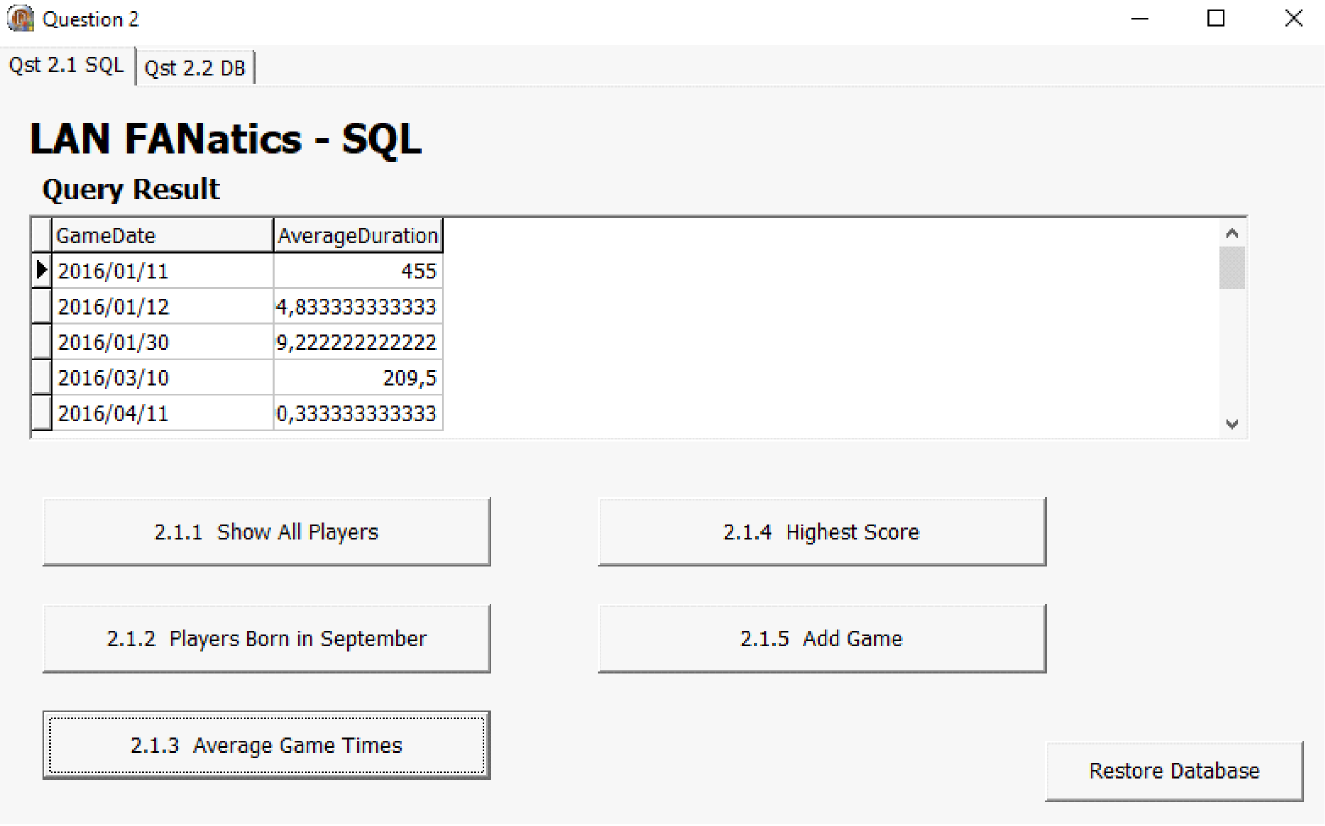
```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

See : https://www.w3schools.com/sql/sql_groupby.asp

We will update our query now to :

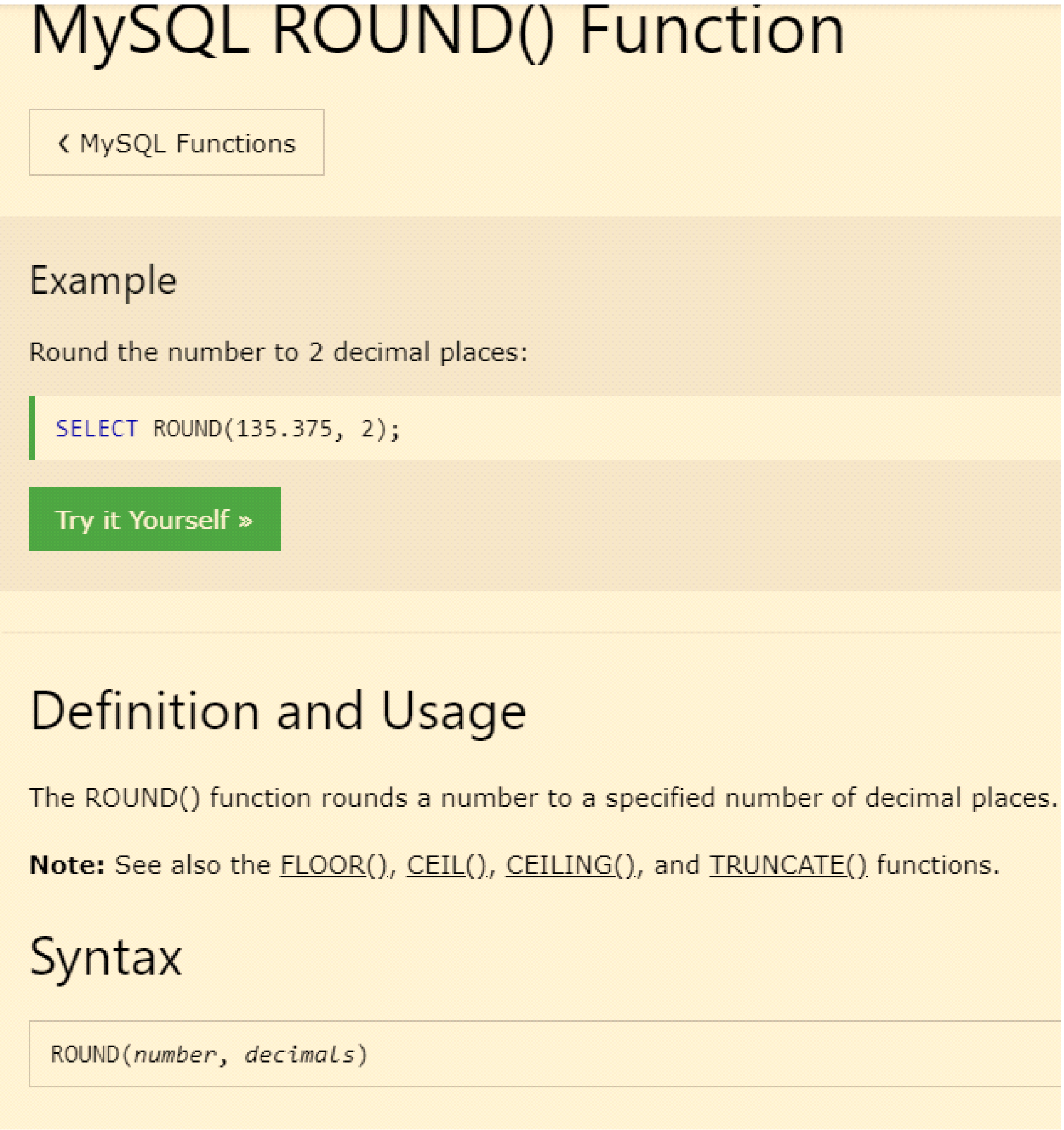
```
procedure TfrmQuestion2.btnSQLqst213Click(Sender: TObject);
begin
   // Question 2.1.3
  sSQL :=('select GameDate,avg(duration) as AverageDuration from tblGames
'+
  'group by GameDate;');

   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```



GameDate	AverageDuration
2016/01/11	455
2016/01/12	4,833333333333
2016/01/30	9,222222222222
2016/03/10	209,5
2016/04/11	0,333333333333

This is not the end of our question, as you can see that the AverageDuration column looks quite unruly, due to a lack of formatting. A requirement of the question, was to Round our output to one decimal place. We will use the ROUND SQL function to do so :



MySQL ROUND() Function

Example

Round the number to 2 decimal places:

```
SELECT ROUND(135.375, 2);
```

Definition and Usage

The ROUND() function rounds a number to a specified number of decimal places.

Note: See also the FLOOR(), CEIL(), CEILING(), and TRUNCATE() functions.

Syntax

```
ROUND(number, decimals)
```

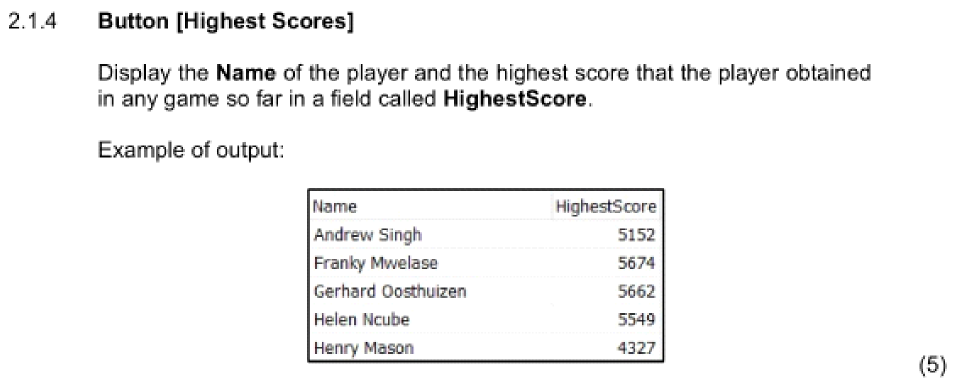
See : https://www.w3schools.com/sql/func_mysql_round.asp

Our final query therefore, will look like this :

```
procedure TfrmQuestion2.btnSQLqst213Click(Sender: TObject);
begin
   // Question 2.1.3
  sSQL :=('select GameDate,Round(avg(duration),1) as AverageDuration from
tblGames '+
  'group by GameDate;');

   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```

Question 2.1.4



2.1.4 **Button [Highest Scores]**

Display the **Name** of the player and the highest score that the player obtained in any game so far in a field called **HighestScore**.

Example of output:

Name	HighestScore
Andrew Singh	5152
Franky Mwelase	5674
Gerhard Oosthuizen	5662
Helen Ncube	5549
Henry Mason	4327

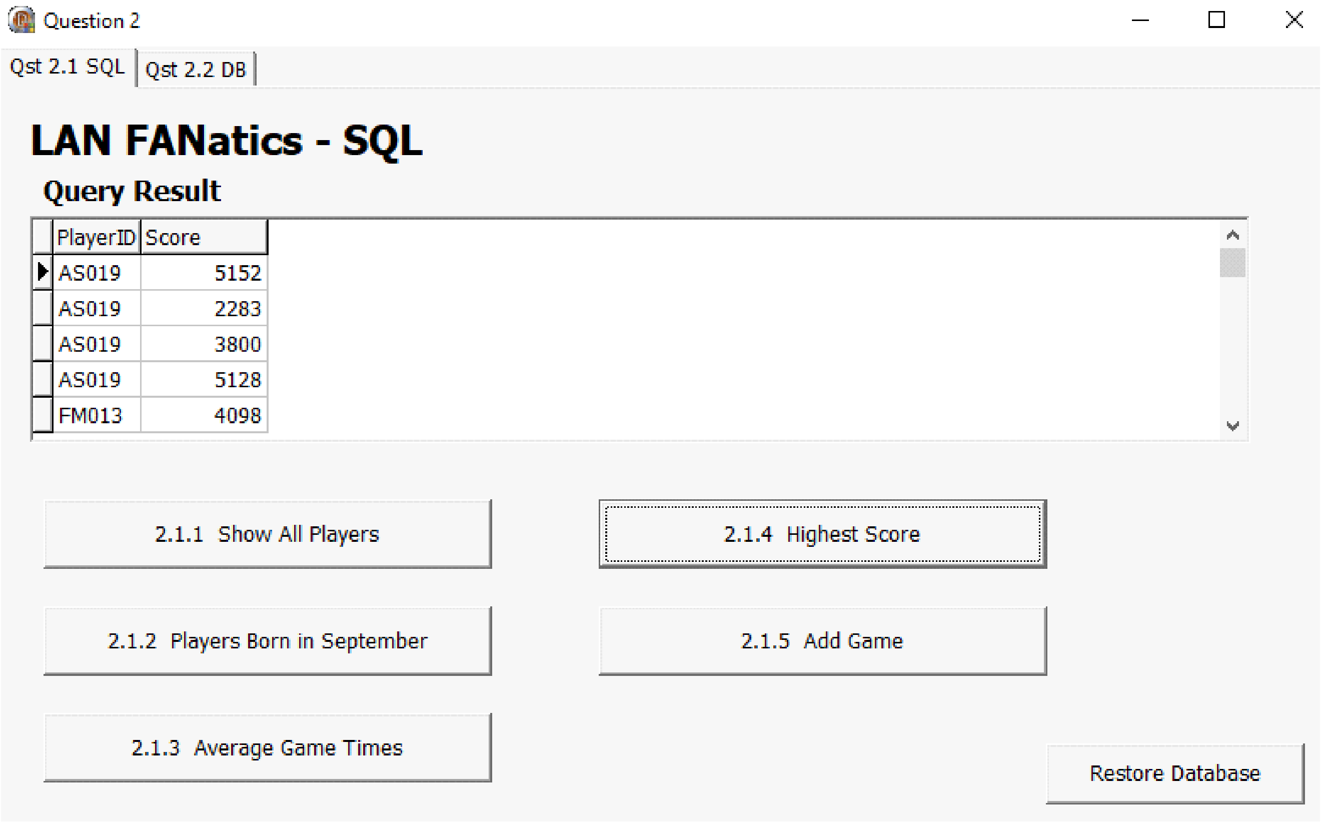
(5)

For this question we have to first understand, from which tables is our data coming from. The question requires us to output two columns Name and Highestscore:

- Name : tblPlayers
- Score : tblGames

We know have to **link** our two tables. How do we do this ? To link the columns we need a common column that is shared between the tables. The common column shared between the two tables is : PlayerID

```
procedure TfrmQuestion2.btnSQLqst214Click(Sender: TObject);
begin
   // Question 2.1.4
   sSQL := 'select tblPlayers.PlayerID,Score from tblPlayers,tblGames '+
   'where tblPlayers.PlayerID = tblGames.PlayerID;';
   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```



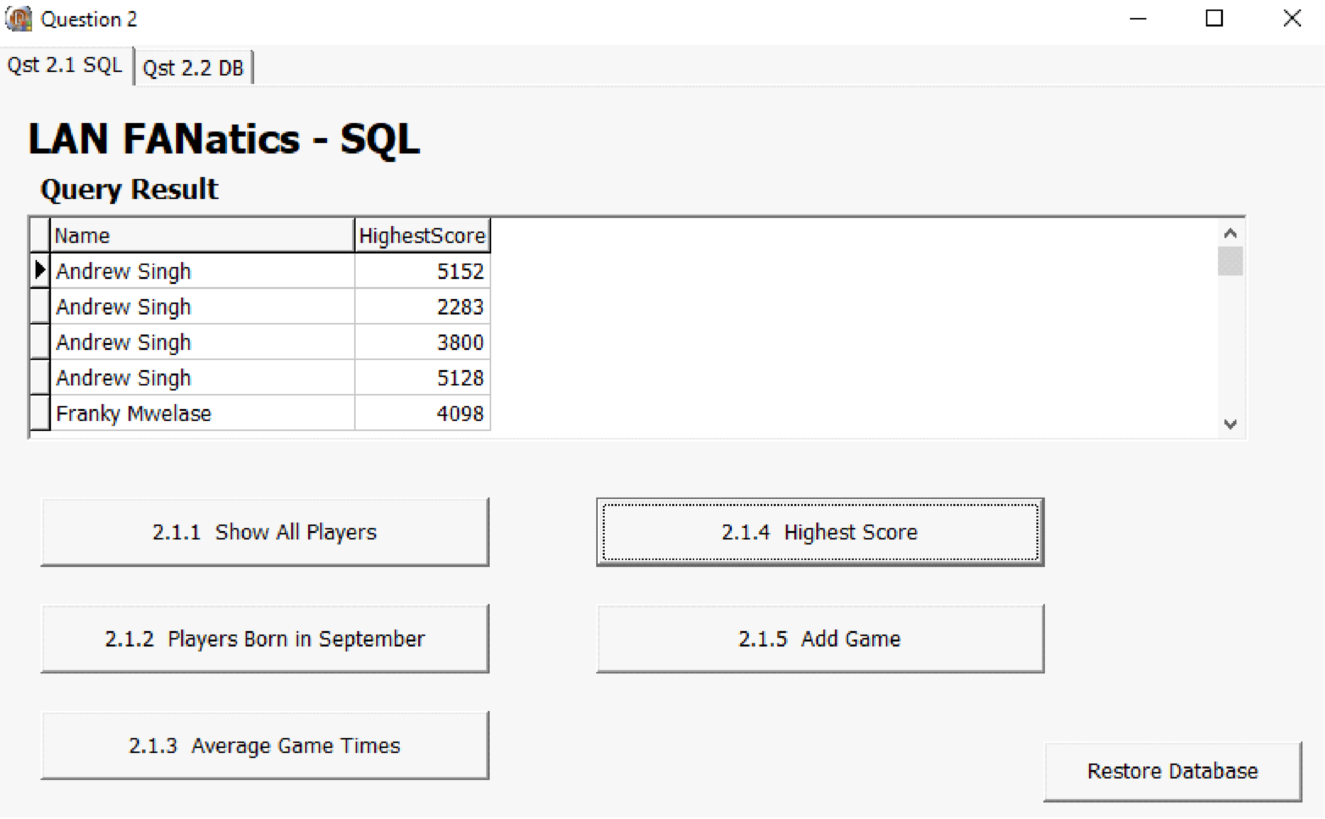
PlayerID	Score
AS019	5152
AS019	2283
AS019	3800
AS019	5128
FM013	4098

The question asks for a Name, and HighestScore columns.

```
procedure TfrmQuestion2.btnSQLqst214Click(Sender: TObject);
```

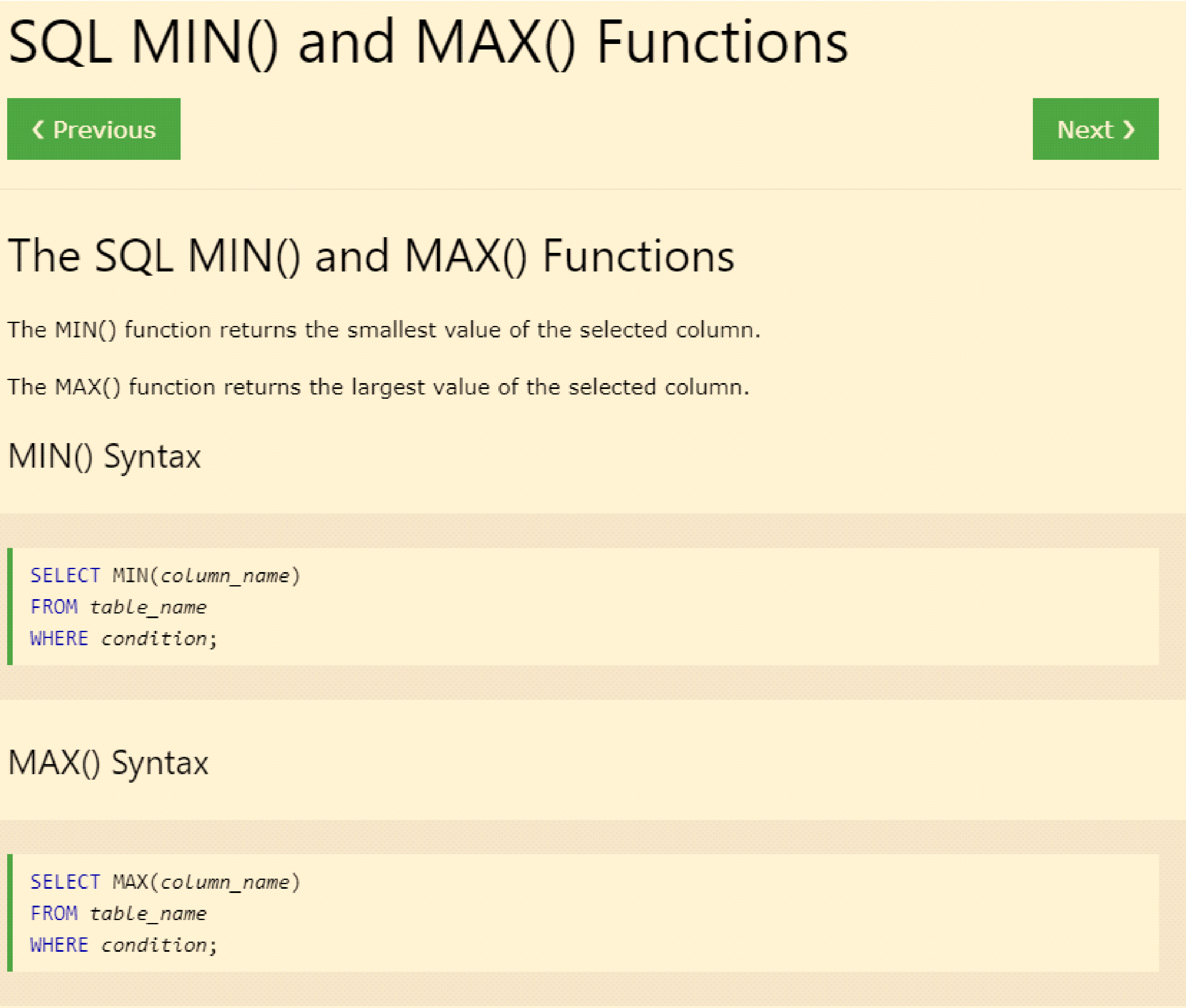
```
begin
   // Question 2.1.4
   sSQL := 'select Name,Score as HighestScore from tblPlayers,tblGames '+
   'where tblPlayers.PlayerID = tblGames.PlayerID;';
   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```

The output for this :



Name	HighestScore
Andrew Singh	5152
Andrew Singh	2283
Andrew Singh	3800
Andrew Singh	5128
Franky Mwelase	4098

Although we are getting the relevant columns, there are duplicates. There the question is how, do we only get the HighestScore? What we are trying to do is, retrieve the Highest Value Or Max from a certain column (HighestScore). Remember, MAX is also an aggregate function.



SQL MIN() and MAX() Functions

The SQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

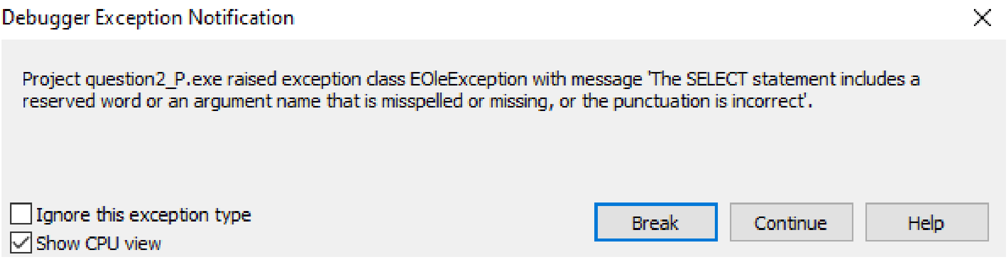
```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

See : https://www.w3schools.com/sql/sql_min_max.asp

If we update our query to :

```
   sSQL := 'select Name,MAX(Score) as HighestScore from
tblPlayers,tblGames '+
   'where tblPlayers.PlayerID = tblGames.PlayerID;';
```

This will however give an error :



Debugger Exception Notification

Project question2_P.exe raised exception class EOleException with message 'The SELECT statement includes a reserved word or an argument name that is misspelled or missing, or the punctuation is incorrect'.

□ Ignore this exception type
☑ Show CPU view

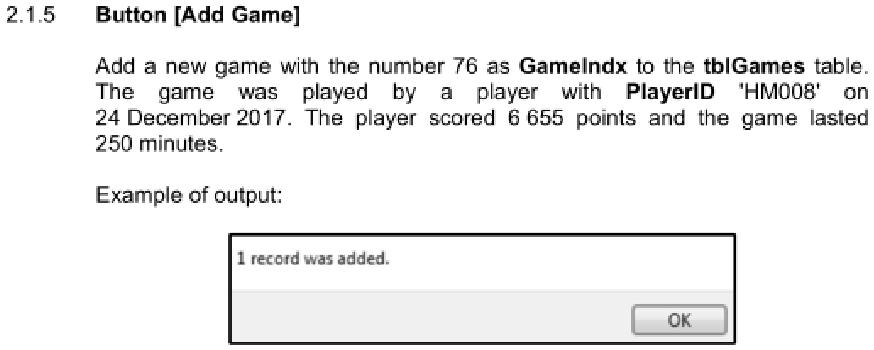
Can you figure out why ? It is because we are using a AGGREGATE function without the GROUP BY Clause.

What are we grouping by ? The Name column.

Our final query will now look like this :

```
procedure TfrmQuestion2.btnSQLqst214Click(Sender: TObject);
begin
   // Question 2.1.4
   sSQL := 'select Name,MAX(Score) as HighestScore from
tblPlayers,tblGames '+
   'where tblPlayers.PlayerID = tblGames.PlayerID GROUP BY Name;';
   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.Open;
end;
```

Question 2.1.5



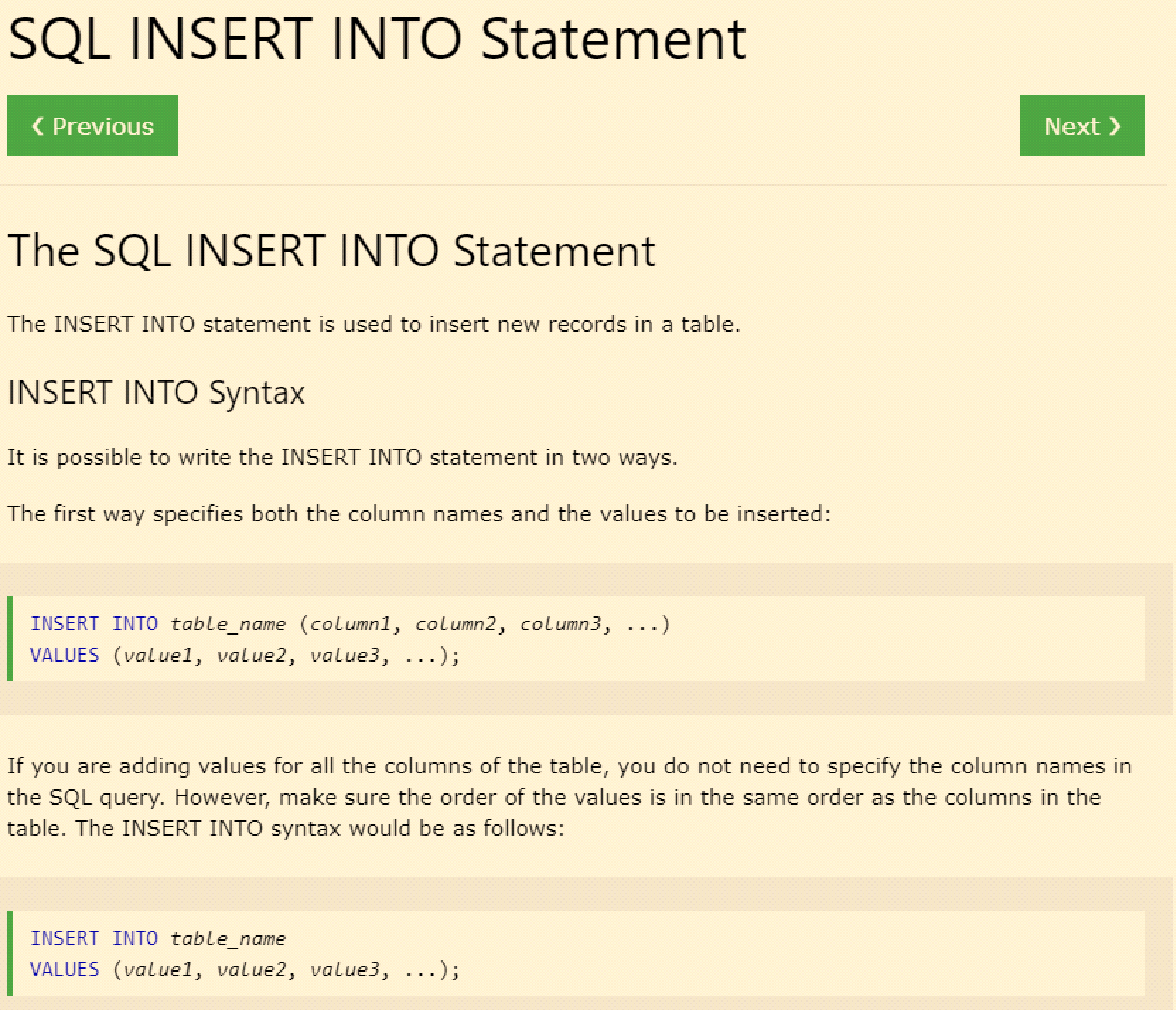
2.1.5 **Button [Add Game]**

Add a new game with the number 76 as **GameIndx** to the **tblGames** table. The game was played by a player with **PlayerID** 'HM008' on 24 December 2017. The player scored 6 655 points and the game lasted 250 minutes.

Example of output:

1 record was added.

Our final question requires us to INSERT a value into the tblGames table.



SQL INSERT INTO Statement

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways.

The first way specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

See : https://www.w3schools.com/sql/sql_insert.asp

First of all what are we inserting ? tblGames has the following columns :

- GameIndx
- GameDate
- PlayerID
- Duartion
- Score

Our corresponding values, based on the question are :

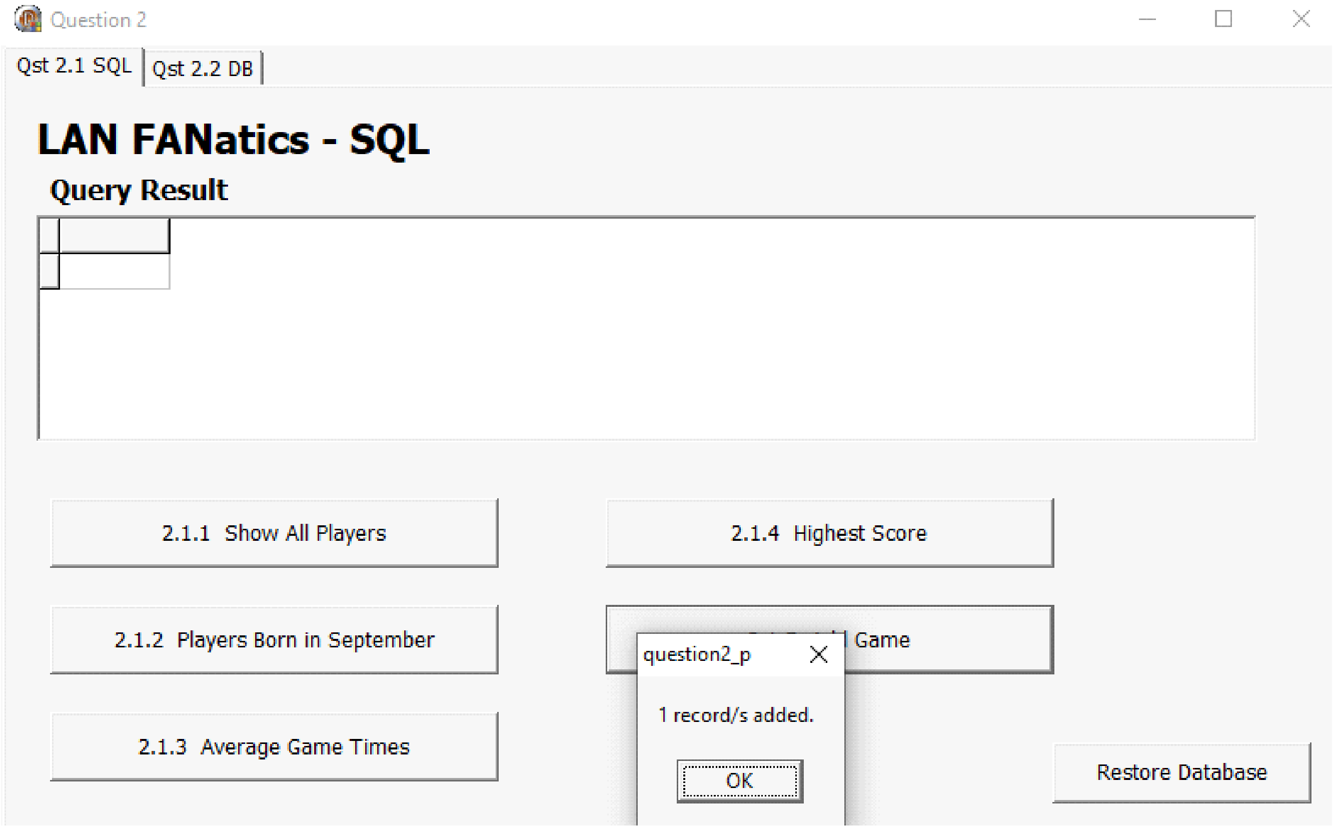
- GameIndx : 76
- GameDate : 24 December 2017
- PlayerID : HM008
- Duartion : 250 minutes
- Score : 6655

We can formulate a INSERT INTO statement :

INSERT INTO tblGames VALUES(76,#24 December 2017#,'HM008',250,6655);

Take note of how the Date 24 December 2017 was typed, the date has to be included in the format #DATE#, and the string value of HM008 was included inside of two apostrophes.''

If we inserted our statement correctly, we will see the output message :



1 record/s added.

Our final code for this question looks like this :

```
procedure TfrmQuestion2.btnSQLqst215Click(Sender: TObject);
begin
   // Question 2.1.5
    sSQL := 'INSERT INTO tblGames VALUES(76,#24 December 2017
#,''HM008'',250,6655);';

   qryLanFans.SQL.Clear;
   qryLanFans.SQL.ADD(sSQL);
   qryLanFans.ExecSQL;
   showMessage(IntToStr(qryLanFans.RowsAffected) + ' record/s added.');
end;
```