SQL Lessons : HockeyDB

Introduction

When dealing with SQL in Delphi, or more specifically understanding SQL to the level expected of you to answer your exam questions, there are few things to note:

1. The examiner/teachers are not cared about any theoretical knowledge pertaining to sql (ie: what is sql ? who invented sql ? why use sql?). Therefore, when learning SQL for your Delphi Exam, you must only pay attention to HOW TO USE SQL(Practical aspect), you can read about all the nice theorectical stuff in your spare time.
2. SQL is similar to learning Maths, and Accounting. There is no much theory. Learning occurrs through practice only.
3. To enable you to practise the SQL commands, I have uploaded the HockeyDB to the rexter website(URLs) given below. You can visit this website and practise the SQL queries.
4. To help you recognize what is sql, and what is not sql in the queries describe below all SQL keywords are typed in Uppercase. Example : SELECT * FROM tblPlayers; (where SELECT and FROM are SQL keywords)
    1. REF: https://www.w3schools.com/sql/sql_ref_keywords.asp
5. We will be referring to W3Schools as reference, please visit the links to help you better understand the sql commands. All references are look like this (Ref : www.url.com)
6. In the exam, the Database will be connected to the program, all you will have to do is write the SQL code.
7. There are four documents that accompany this document which you find in the sql folder :
    1. hockeydb_exam_question.pdf
    2. hockeydb_exam_memo.pdf
    3. dancedb_exam_question.pdf
    4. dancedb_exam_memo

The HockeyDB is uploaded here, at the ***REXTER PLAYGROUND*** where you will be practising your SQL:

```
Hockey Database :
    https://rextester.com/WYWYR34218

Dance Database :
    https://rextester.com/OHDRR81082
```

(Note: this link is not permenent. If you are not seeing the desired results, contact me I will reupload)

In this document, I run SQL commands like this :

```
mysql> SHOW TABLES;
```

Where ANTYHING after "mysql>" is the SQL command. You can take that SQL command, and paste into the rexter URL, and run it yourself! And when you are comforatable with SQL, you can put the command into Delphi and run it there.

Database

When you are presented with a SQL exam question these are the first steps you need to take :

(1). Look at the table, and examine all the fields(columns). At the back of the exam paper, there will be a seperate table describing the information contained in the database tables(ie: the data types of the columns). Open up hockeydb_exam_question.pdf and scroll right to end, you will see what I mean.

To help us achieve the first step listed above, we will use the :

```
    * DESCRIBE :    shows the tables properties and data types in a given
database(HockeyDB in this case)
    * SHOW TABLES : shows the different tables in a given database
    * SELECT * FROM tblame : translated into english means SELECT ALL FROM
tablename(selects all columns and returns the corresponding records(rows))
```

mysql> DESCRIBE tblTeams;

```
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| TeamName           | varchar(6)  | YES  |     | NULL    |       |
| Coach              | varchar(11) | YES  |     | NULL    |       |
| NumberOfGamesPlayed | tinyint(4)  | YES  |     | NULL    |       |
| NumberOfGamesWon   | tinyint(4)  | YES  |     | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
```

mysql> DESCRIBE tblPlayers;

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| PlayerID     | tinyint(4)  | YES  |     | NULL    |       |
| PlayerSurname | varchar(10) | YES  |     | NULL    |       |
| PlayerName   | varchar(11) | YES  |     | NULL    |       |
| IDNumber     | bigint(20)  | YES  |     | NULL    |       |
| TeamName     | varchar(6)  | YES  |     | NULL    |       |
| SkillsLevel  | tinyint(4)  | YES  |     | NULL    |       |
| GoalKeeper   | varchar(5)  | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

mysql> SHOW TABLES;

```
+------------------+
| Tables_in_hockey |
+------------------+
| CopytblPlayers   |
| CopytblTeams     |
| tblPlayers       |
| tblTeams         |
+------------------+
```

mysql> SELECT * FROM tblPlayers;

```
+----------+--------------+-------------+-------------+----------+-------
------+------------+
| PlayerID | PlayerSurname | PlayerName  | IDNumber     | TeamName |
SkillsLevel | GoalKeeper |
+----------+--------------+-------------+-------------+----------+-------
------+------------+
|        1 | Fivaz         | Peter       | 306170790504 | u/16 A   |
6 | False      |
|        2 | Maphaphu      | Bukelwa     | 609220225852 | u/14 B   |
3 | False      |
|        3 | Snyders       | Con         | 602200518279 | u/14 B   |
3 | False      |
|        4 | Jagers        | Ben         | 403060227644 | u/16 B   |
4 | True       |
|        5 | Plaatjie      | Zolile      | 105100314587 | u/18 B   |
4 | False      |
```

mysql> SELECT * FROM tblTeams;

```
+----------+-------------+---------------------+------------------+
| TeamName | Coach       | NumberOfGamesPlayed | NumberOfGamesWon |
+----------+-------------+---------------------+------------------+
| u/14 A   | Modikaze, P |                   8 |                4 |
| u/14 B   | Smith, GP   |                   6 |                3 |
| u/16 A   | Smit, J     |                   8 |                7 |
| u/16 B   | Mpofu, XB   |                   6 |                3 |
| u/18 A   | Decan, H    |                  10 |                8 |
| u/18 B   | Mullan, NV  |                   5 |                1 |
| u/18 C   | Botha, M    |                   4 |                1 |
+----------+-------------+---------------------+------------------+
```

SQL Syntax

When practising SQL on Rexter, you can paste a statement in like this :

```
SELECT * FROM tblTeams;
```

However in Delphi, the same statement will look this:

```
procedure TfrmDBQuestion2.btnQ2_1_1Click(Sender: TObject);
var
sSQL1: String;
begin // Question 2.1.1
sSQL1 := 'SELECT * FROM tblTeams';

// Provided code
dbCONN.RunSQL(sSQL1);
end;
```

Or it can look like this:

```
procedure TfrmRec.btnAClick(Sender: TObject);
begin
qryRec.Active := False;
qryRec.SQL.Text :=  'SELECT * FROM tblTeams';
qryRec.Active := True;
end;
```

Do not stress, the teachers make it VERY OBVIOUS where you must your SQL code. I only mention it, so that you are aware, of how delphi expects SQL as oppose to the Rexter playground.

Its worth noting that SQL statements are not case-senstive, and table names are case-sensitive as seen below, where the two select statements are equivalent, and ***tblPlayers*** is the table name.

```
select playername from tblPlayers;
```

Can be typed as :

```
SELECT PLAYERNAME FROM tblPlayers;
```

Connecting A .mdb Database To Delphi

This section is not relevant for the exam, as the connection is part of the question/code provided. This section is only relevant for your IT PAT 2020 assignment, where you have to create the database connection yourself. **Note** Sample Program containing what is described below located at : ***hockey_connection***

1. Make sure your .mdb file is present in the same directory as your project files.

2. From the Tool Palette on the right hand side of the screen:

    1. Add a TADOQuery to the form. Then under the Object inspector in the left side of the screen, with the TADOQuery selected click on CONNECTION STRING. You will three ellipsis dots(...), click it to build the CONNECTION STRING
        1. Click Build
        2. Select Microsoft Jet 4.0 OLE DB Provider, click Next
        3. Under "Select or enter a database name:", click the ellipsis dots(...) on the right an select your database file(.mdb)
            1. The datababase string will look like this "E:\sql\hockey_connection\HockeyDB.mdb" -> This is a abosolute file path name
                1. Remove everything from the start of the string, up until the name of the database. So the above path becomes "HockeyDB.mdb" which is a relative path name. The reason we did is because when you move your project to another a computer, this location might not be found "E:\sql\hockey_connection", therefore by specifying only "Hockey.mdb", we tell the program that the database is in the current folder, so irrespective where the project folder moves to, it knows the database file is located in side of it. * Final note when typing in the strings do not include the "".
            2. Under Enter information to log on to the database:
            3. Username and Password Must be empty
            4. Blank Password and Allowing Saving Password Checkboxes must be empty
        4. Click Test Connection to make sure your Database is connected Properly. You should see "Test connection succeeded."
        5. Then press Ok at the bottom.
        6. Then press ok again, you should see something like "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=HockeyDB.mdb;Persist Security Info=False" under Use Connection string.

3. From the Tool Palette on the right side:

    1. Add a DATASOURCE to the form. Then under the Object inspector in the left side of the screen, with DATASOURCE selected click DATASET and set it ADOQUERY1. So it looks like this : DATASET=ADOQUERY1.

4. From the Tool Palette on the right side:

    1. Add a TDBGrid to the form. Then under the Object inspector in the left side of the screen, with TDBGrid selected, set the DATASOURCE=Datasource1

5. Finally, to demontrate it is working, add a TButton to the form, double click it and you add the following code and run the program.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Adoquery1.Close;
Adoquery1.SQL.Text := 'select * from tblPlayers';
adoquery1.Open;
end;
```

Connecting a .mdb Database to Delphi and Reading Table Information into an array

Here will demonstrate how to read your .mdb database into a Delphi Class. (This section also not relevant to exams - only relevant to PAT assignment)

As part of the IT PAT 2020 assignment, your "data" is supposed to be stored in a .mdb database. You will then use SQL to insert/delete/update and run other queries as specified in the requirements. However, as part IT PAT 2020 aswell, it is required that you use Objects and Classes & TextFiles.

Note Sample Program containing what is described below located at : ***hockey_class***

1. Start a new project (or in the case of your PAT you will be adding a new form or using an existing an existing one.
2. From the Tool Palette on the right hand side:
    1. Add a TADOConnection to the form. Then under the Object inspector in the left side of the screen, with the TADOconnection selected click on CONNECTION STRING. You will three ellipsis dots(...), click it to build the CONNECTION STRING
        1. Click Build
        2. Select Microsoft Jet 4.0 OLE DB Provider, click Next
        3. Under "Select or enter a database name:", click the ellipsis dots(...) on the right an select your database file(.mdb)
            1. The datababase string will look like this "E:\sql\hockey_connection\HockeyDB.mdb" -> This is a abosolute file path name
                1. Remove everything from the start of the string, up until the name of the database. So the above path becomes "HockeyDB.mdb" which is a relative path name. The reason we did is because when you move your project to another a computer, this location might not be found "E:\sql\hockey_connection", therefore by specifying only "Hockey.mdb", we tell the program that the database is in the current folder, so irrespective where the project folder moves to, it knows the database file is located in side of it. * Final note when typing in the strings do not include the "".
            2. Under Enter information to log on to the database:
            3. Username and Password Must be empty
            4. Blank Password and Allowing Saving Password Checkboxes must be empty
        4. Click Test Connection to make sure your Database is connected Properly. You should see "Test connection succeeded."
        5. Then press Ok at the bottom.
        6. Then press ok again, you should see something like "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=HockeyDB.mdb;Persist Security Info=False" under Use Connection string.
        7. Make sure Login Prompt Property is Unchecked.
3. From the Tool Palette on the right hand side:

    1. Add a TADOTable to the form. Then under the Object insepector in the left side of the screen, with the TADOTable selected click on CONNECTION=TADOConncetion1
    2. Then under the Object insepector in the left side of the screen, with the TADOTable selected click TABLENAME=tblTeams.
    3. Then under the Object insepector in the left side of the screen, with the TADOTable selected click Active=True (its a checkbox)

4. From the Tool Palette on the right hand side:

    1. Add a TRichEedit to the form.

5. From the Tool Palette on the right hand side:

    1. Add a TButton to the form.

6. With the Form selected, Click on Swith from Design View to Code View.

7. We will now declare a new class, that will hold the data from tblTeams.

    1. Under the type keyword enter the follwing:

```
  TTeams = class(TObject)
      private
      sTeamName : string;
      sCoach : string;
      iNumberOfGamesPlayed : integer;
      iNumberOfGamesWon : integer;
      public
      constructor
create(teamname:string;coach:string;gamesplayed:integer;gameswon:
integer);
      function tostring: string;
      function getNumberOfGamesPlayed : integer;
      function getNUmberOfGamesWon : integer;
      function getTeamsName : string;
      function getCoach : string;
  end;
```

    If you are confused about what is going in this class, use these references to find out more:

    Ref : Procedure vs Functions : http://www.delphibasics.co.uk/Article.asp?Name=Routines

    Ref : Classes : http://www.delphibasics.co.uk/RTL.asp?Name=Class

    Ref : Classes: http://www.teachitza.com/delphi/exampleclass.htm

    Ref : Constructor : http://www.delphibasics.co.uk/RTL.asp?Name=Constructor

    Note: The teachitza.com is a great reference, you will other valuable delphi programming resources on this website.

8. Then under implementaion keyword, add the following implementation :

```
 constructor TTeams.create(teamname: string; coach: string;
gamesplayed: Integer; gameswon: Integer);
 begin
     sTeamname := teamname;
     sCoach := coach;
     iNumberOfGamesPlayed := gamesplayed;
     iNumberOfGamesWon := gameswon;
 end;

 function TTeams.tostring;
 begin
     Result := sTeamname + #9 + sCoach + #9 +
inttostr(iNumberOfGamesPlayed) + #9 + inttostr(iNumberOfGamesWon);
 end;

 function TTeams.getCoach;
 begin
     Result := sCoach;
 end;

 function TTeams.getTeamsName;
 begin
     Result := sTeamName;
 end;

 function TTeams.getNUmberOfGamesWon;
 begin
     Result := iNumberOfGamesWon;
 end;

 function TTeams.getNumberOfGamesPlayed;
 begin
     Result := iNumberOfGamesPlayed;
 end;
```

9. Our class is now delcared, now we can read in our data from our tblTeams(located in the .mdb database) into a data strucure(class/array) so that we can manipulate the table data within the program. Now we declare a GLOBAL array that will store all the "Team". Under var keyword add, aswell as a count integer variable :

```
 var
     Form1: TForm1;
     arrTeams : array[1..50] of TTeams; // you will add this line
     iCount : integer; //  add this line also - keep track of the
items in our array
```

10. Go Back to the Form Gui, Double click on the Button1 to open up the onclick procedure for that button.

```
procedure TForm1.Button1Click(Sender: TObject);
var sTeamName,sCoach : string;
    iGamesWon,iGamesPlayed,i: integer;
begin
    ADOtable1.First; // point to the first record in tbltTeams
    iCount := 0;
    while not ADOtable1.Eof do  // while not tblTeams of EndOfFile
    begin
    inc(iCount);
    sTeamName := ADOtable1['TeamName'];
    sCoach := ADOtable1['Coach'];
    iGamesWon := ADOtable1['NumberOfGamesWon'];
    iGamesPlayed := ADOtable1['NumberOfGamesPlayed'];

    arrTeams[iCount] :=
TTeams.create(sTeamName,sCoach,iGamesPlayed,iGamesWon);
    ADOtable1.Next; // point to the next entry in tblTeams
    end;

    showmessage('iCount='+inttostr(icount));
    // Print all the data from the array
    RichEdit1.Clear;
    RichEdit1.Paragraph.TabCount := 5;

    RichEdit1.Paragraph.Tab[0] := 100;
    RichEdit1.Paragraph.Tab[1] := 200;
    RichEdit1.Paragraph.Tab[2] := 300;
    RichEdit1.Paragraph.Tab[3] := 400;

    for i := 1 to iCount do
    begin
        RichEdit1.Lines.Add(arrTeams[i].tostring);
    end;


end;
```

11. Run the program a and tblTeams information will be printed to the RichEdit.

SQL Statements

Go to URL to test these statements:

```
https://rextester.com/OHDRR81082

The DanceCompetition database contains two tables:

mysql > DESCRIBE tblDanceCouples;

+----------------------+-------------+------+-----+---------+-------+
```

```
| Field                 | Type        | Null | Key | Default | Extra |
+-----------------------+-------------+------+-----+---------+-------+
| DanceCoupleID         | int(11)     | YES  |     | NULL    |       |
| DancePartner1         | varchar(20) | YES  |     | NULL    |       |
| DancePartner2         | varchar(20) | YES  |     | NULL    |       |
| ProfessionalDancers   | varchar(2)  | YES  |     | NULL    |       |
| PreviousDancePartners | char(1)     | NO   |     | NULL    |       |
+-----------------------+-------------+------+-----+---------+-------+

mysql > DESCRIBE tblResults;

+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| RoutineNo     | int(11)     | YES  |     | NULL    |       |
| Week          | int(11)     | YES  |     | NULL    |       |
| Round         | int(11)     | YES  |     | NULL    |       |
| DanceCoupleID | int(11)     | YES  |     | NULL    |       |
| TypeOfDance   | varchar(60) | YES  |     | NULL    |       |
| Song          | varchar(80) | YES  |     | NULL    |       |
| Score         | int(11)     | YES  |     | NULL    |       |
| Result        | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------
```

Select

The select statement retrieves data from the table, based on the column you specify. The syntax for this statement: SELECT column FROM table;

```
mysql> SELECT song from tblResults;
+----------------------------------------+
| song                                   |
+----------------------------------------+
| Who's That Chick?                      |
| Are You Lonesome Tonight?              |
| I've Got The Music in Me               |
| Angel                                  |
| Bad Boys                               |
| Three Times a Lady                     |
| Venus                                  |
```

If You wanted to select more than one column, you can add more columns seperated by commas:

```
mysql> SELECT TypeOfDance,song,Score from tblResults;
+-----------------+----------------------------------------+-------+
| TypeOfDance     | song                                   | Score |
+-----------------+----------------------------------------+-------+
| Cha-cha         | Who's That Chick?                      |    28 |
| Waltz           | Are You Lonesome Tonight?              |    24 |
```

```
| Cha-cha          | I've Got The Music in Me                    |    17 |
| Waltz            | Angel                                       |    20 |
| Cha-cha          | Bad Boys                                    |    19 |
```

If you wanted to select the entire table, you can use the asterisk as a wildcard:

```
mysql> SELECT * from tblResults;
+-----------+------+-------+--------------+-----------------+-------------
-----------------------------+-------+------------+
| RoutineNo | Week | Round | DanceCoupleID | TypeOfDance     | Song
| Score | Result     |
+-----------+------+-------+--------------+-----------------+-------------
-----------------------------+-------+------------+
|         1 |    1 |     1 |             1 | Cha-cha         | Who's That
Chick?                          |    28 | Safe       |
|         2 |    1 |     1 |             2 | Waltz           | Are You
Lonesome Tonight?                |    24 | Safe       |
|         3 |    1 |     1 |             3 | Cha-cha         | I've Got The
Music in Me                   |    17 | Safe       |
|         4 |    1 |     1 |             4 | Waltz           | Angel
|    20 | Safe       |
|         5 |    1 |     1 |             5 | Cha-cha         | Bad Boys
```

Where

The Where clause allows us to specify criteria, to narrow down our results to what we want. The syntax for this statement: SELECT column FROM table WHERE criteria;In the below code, we select two Columns(DanceCoupleID,Result) from tblResults where Result is Eliminated. This could have been rephrased into a question, stating ‘show all the DanceCoupleIDs who have been eliminated.’

```
mysql> SELECT DanceCoupleID,Result FROM tblResults WHERE Result =
'Eliminated';
+---------------+------------+
| DanceCoupleID | Result     |
+---------------+------------+
|            12 | Eliminated |
|             2 | Eliminated |
|             9 | Eliminated |
|            13 | Eliminated |
|             3 | Eliminated |
|             4 | Eliminated |
```

Aggregate Functions

An aggregate function is one which groups the values of multiple rows, into a single value of more meaningful information. Aggreate functions you will work with include, Min, Max, Count, Sum, Avg, Count

MIN Returns the Minimum value from a column of values.

```
mysql> SELECT MIN(Score) from tblResults;
+------------+
| MIN(Score) |
+------------+
|         12 |
+------------+
```

MAX Returns the Maximum value from a column of values.

```
mysql> SELECT MAX(Score) from tblResults;
+------------+
| MAX(Score) |
+------------+
|         40 |
+------------+
```

SUM Returns the SUM from a column of values.

```
mysql> SELECT SUM(Score) from tblResults;
+------------+
| SUM(Score) |
+------------+
|       3523 |
+------------+
```

AVG

Returns the Average from a column of values.

```
mysql> SELECT AVG(Score) from tblResults;
+------------+
| AVG(Score) |
+------------+
|    29.6050 |
+------------+

mysql> SELECT FORMAT(AVG(Score),2) from tblResults;
+----------------------+
| FORMAT(AVG(Score),2) |
+----------------------+
| 29.61                |
+----------------------+
```

Count

```
mysql> SELECT Count(Score) from tblResults;
+--------------+
| Count(Score) |
+--------------+
|          119 |
+--------------+
```

Counts the number of Values that match a certain Criteria.

ORDER BY

We use the Order By clause to arrange our results in alphabetical order. When we do not specify the Order By clause, results are returned from the table in which order they found. You can specify ascending or descending order. The syntax for for this statement:

```
SELECT column FROM table WHERE criteria Order By column
```

In the below code, we select two columns(DanceCoupleID,TypeOfDance) from tblResults:

```
mysql> SELECT DanceCoupleID,TypeOfDance from tblResults where result =
'Safe';
+---------------+-----------------+
| DanceCoupleID | TypeOfDance     |
+---------------+-----------------+
|             1 | Cha-cha         |
|             2 | Waltz           |
|             3 | Cha-cha         |
|             4 | Waltz           |
|             5 | Cha-cha         |
|             6 | Waltz           |
```

Now we use the ORDER BY clause ascending:

mysql> SELECT DanceCoupleID,TypeOfDance from tblResults where result = 'Safe' ORDER BY TypeOfDance asc; +---------------+-----------------+ | DanceCoupleID | TypeOfDance | +---------------+-----------------+ | 6 | American Smooth | | 1 | American Smooth | | 5 | American Smooth | | 7 | American Smooth | | 10 | American Smooth | | 14 | American Smooth |

Now we use the ORDER BY clause descending:

```
mysql> SELECT DanceCoupleID,TypeOfDance from tblResults where result =
'Safe' ORDER BY TypeOfDance desc;
+---------------+-----------------+
| DanceCoupleID | TypeOfDance     |
+---------------+-----------------+
|             2 | Waltz           |
```

```
|             4 | Waltz           |
|             6 | Waltz           |
|             9 | Waltz           |
|            11 | Waltz           |
|            13 | Waltz           |
```

GROUP BY

The Group By statement is used when you have the repeating values occuring in Columns. It groups these multiple occurence of values, into disctinct values. It is often used with aggregate functions (MAX,MIN,SUM,COUNT,AVG). The syntax for this statement is:

```
SELECT columns from table WHERE condition GROUP BY column.
```

To better undestand the Group By statement, lets look at the data from tblResults, where all 4 of the columns selected have repeating values in them:

```
mysql> select Week,Round,DanceCoupleID,TypeOfDance from tblResults;
+------+-------+---------------+-----------------+
| Week | Round | DanceCoupleID | TypeOfDance     |
+------+-------+---------------+-----------------+
|    1 |     1 |             1 | Cha-cha         |
|    1 |     1 |             2 | Waltz           |
|    1 |     1 |             3 | Cha-cha         |
|    1 |     1 |             4 | Waltz           |
|    1 |     1 |             5 | Cha-cha         |
|    1 |     1 |             6 | Waltz           |
```

If we wanted to know how many times the Cha-Cha was danced per Week:

```
mysql> SELECT Week,Count(TypeOfDance),TypeOfDance from tblResults where
TypeOfDance='Cha-Cha' GROUP BY Week;
+------+--------------------+-------------+
| Week | Count(TypeOfDance) | TypeOfDance |
+------+--------------------+-------------+
|    1 |                  8 | Cha-cha     |
|    3 |                  1 | Cha-cha     |
|    4 |                  1 | Cha-cha     |
|    7 |                  1 | Cha-cha     |
|    9 |                  1 | Cha-cha     |
+------+--------------------+-------------+
```

If we wanted to know how many couples where Eliminated per Week:

```
mysql> SELECT Week,Count(Result) from tblResults WHERE Result='Eliminated'
GROUP BY Week;
+------+---------------+
| Week | Count(Result) |
+------+---------------+
|    2 |             1 |
|    3 |             1 |
|    4 |             1 |
|    5 |             1 |
|    6 |             1 |
|    7 |             1 |
|    8 |             1 |
|    9 |             1 |
|   10 |             1 |
|   11 |             2 |
|   12 |             2 |
+------+---------------+
```

If we wanted to know which TypeOfDance yieled the highest average score:

```
mysql> SELECT TypeOfDance,AVG(Score) FROM tblResults GROUP BY TypeOfDance
ORDER BY AVG(Score) desc;
+-----------------+------------+
| TypeOfDance     | AVG(Score) |
+-----------------+------------+
| Showdance       |    37.6667 |
| Argentine Tango |    36.2000 |
| Charleston      |    34.2000 |
| Quickstep       |    33.8000 |
| American Smooth |    33.7500 |
| Jive            |    30.8000 |
| Tango           |    30.6364 |
| Viennese Waltz  |    29.6000 |
| Rumba           |    28.8571 |
| Samba           |    28.7143 |
| Foxtrot         |    28.5000 |
| Paso Doble      |    27.5714 |
| Waltz           |    26.2222 |
| Salsa           |    25.2000 |
| Cha-cha         |    24.0000 |
+-----------------+------------+
```

IN

The IN clause allows us to state which values we would like to see for a certain column. The syntax for this statement is:

```
SELECT column from table WHERE column IN('value1','value2');
```

Lets say we wanted to find out all the DanceCoupleIDs that used the Cha-Cha Dance, and Salsa Dance.

```
mysql> SELECT DanceCoupleID,TypeOfDance from tblResults WHERE TypeOfDance
IN('Cha-Cha','Salsa');
+---------------+-------------+
| DanceCoupleID | TypeOfDance |
+---------------+-------------+
|             1 | Cha-cha     |
|             3 | Cha-cha     |
|             5 | Cha-cha     |
```

LIKE

The Like operator allows us to search for a matching pattern within a column. The syntax for this statement is: SELECT column FROM table WHERE column LIKE pattern It works alongside two wildcards:

```
% - The percent sign represents zero, one, or multiple characters
_ - The underscore represents a single character
```

If we wanted to find out all the Songs that start with the letter B:

```
mysql> SELECT Song from tblResults where SONG LIKE('B%');
+--------------------------+
| Song                     |
+--------------------------+
| Bad Boys                 |
| Build Me Up Buttercup    |
| Buona Sera               |
| Breakeven                |
| Be Italian               |
```

If we wanted to know all the songs that contain the word ‘Love’ within them:

```
mysql> SELECT Song from tblResults where SONG LIKE('%LOVE%');
+---------------------------+
| Song                      |
+---------------------------+
| When Love Takes Over      |
| Papa Loves Mambo          |
| Somebody to Love          |
| Love Man                  |
```

If we wanted to known those RoutineNos that obtained a score of 30 or greater:

```
mysql> SELECT RoutineNo,Score from tblResults WHERE Score LIKE('3_');
+-----------+-------+
| RoutineNo | Score |
+-----------+-------+
|        14 |    32 |
|        21 |    30 |
|        25 |    33 |
|        29 |    30 |
|        32 |    32 |
```

HAVING

The Having statement is similar to the where statement, allowing columns to be filtered based on a criteria, but with a few differences. The HAVING statement is applied after the GROUP BY Statement, whilst the WHILE statement is applied before the GROUP BY. Another important difference is that the HAVING STATEMENT can filter on aggregrate results. The syntax for this statement is as follows:

```
SELECT column FROM table_name WHERE condition GROUP BY column HAVING
condition;
```

To understand the HAVING statement better we will use the explain keyword:

```
mysql> EXPLAIN SELECT Week,Round FROM tblResults HAVING Week < 2 LIMIT 2;
+----+-------------+-----------+-----------+------+---------------+------
+---------+------+------+----------+-------+
| id | select_type | table      | partitions | type | possible_keys | key
| key_len | ref  | rows | filtered | Extra |
+----+-------------+-----------+-----------+------+---------------+------
+---------+------+------+----------+-------+
|  1 | SIMPLE      | tblResults | NULL       | ALL  | NULL          | NULL
| NULL    | NULL |  118 |   100.00 | NULL  |
+----+-------------+-----------+-----------+------+---------------+-----

mysql> EXPLAIN SELECT Week,Round FROM tblResults WHERE Week < 2 LIMIT 2;
+----+-------------+-----------+-----------+------+---------------+------
+---------+------+------+----------+-------------+
| id | select_type | table      | partitions | type | possible_keys | key
| key_len | ref  | rows | filtered | Extra       |
+----+-------------+-----------+-----------+------+---------------+------
+---------+------+------+----------+-------------+
|  1 | SIMPLE      | tblResults | NULL       | ALL  | NULL          | NULL
| NULL    | NULL |  118 |    33.33 | Using where |
+----+-------------+-----------+-----------+------+---------------+------
+---------+------+------+-
```

In the above code there is a filtered column. This column shows a percentage of how many rows where filtered. As seen with the HAVING statement 100% of the columns where filtered, meaning the criteria Week < 2 was applied to the entire table before the results where returned. Where as with the While Statement only 33.33% of the Rows where filtered against the statement Week < 2. This implies that the WHILE statement is more efficient than the HAVING statement.

However, the HAVING statement can do something the WHERE cannot- and that is to filter aggregate results. The below statement tells us what was the max score for each Dance:

```
mysql> SELECT TypeOfDance,MAX(Score) FROM tblResults GROUP BY TypeOfDance;
+-----------------+------------+
| TypeOfDance     | MAX(Score) |
+-----------------+------------+
| American Smooth |         39 |
| Argentine Tango |         40 |
| Cha-cha         |         32 |
| Charleston      |         39 |
| Foxtrot         |         36 |
| Jive            |         39 |
```

But we can refine it futher by searching for max Scores greater than 35.

```
mysql> SELECT TypeOfDance,MAX(Score) FROM tblResults GROUP BY TypeOfDance
HAVING MAX(Score) > 35;
+-----------------+------------+
| TypeOfDance     | MAX(Score) |
+-----------------+------------+
| American Smooth |         39 |
| Argentine Tango |         40 |
| Charleston      |         39 |
| Foxtrot         |         36 |
| Jive            |         39 |
```

INSERT

The INSERT statement allows us to add a row into a table. The syntax for this statement is as follows:

```
INSERT INTO table(column1,column2) VALUES(value1,value2);

mysql> INSERT INTO
tblResults(RoutineNo,Week,Round,DanceCoupleID,TypeOfDance,Song,Score,Result
)
   VALUES('120','13','2','15','Classical','Symphony 5','40','Eliminated');
```

UPDATE

The UPDATE statement allows us to update a field value within a table. The syntax for this statement is as follows:

```
UPDATE table SET column WHERE condition;

mysql> UPDATE tblResults SET TypeOfDance='Swag' WHERE RoutineNo='120';
```

DELETE

The DELETE statement will delete a row within a table. The syntax for this statement is as follows:

```
DELETE FROM table WHERE condition;

mysql> DELETE FROM tblResults WHERE TypeOfDance LIKE('%Swag%');
```