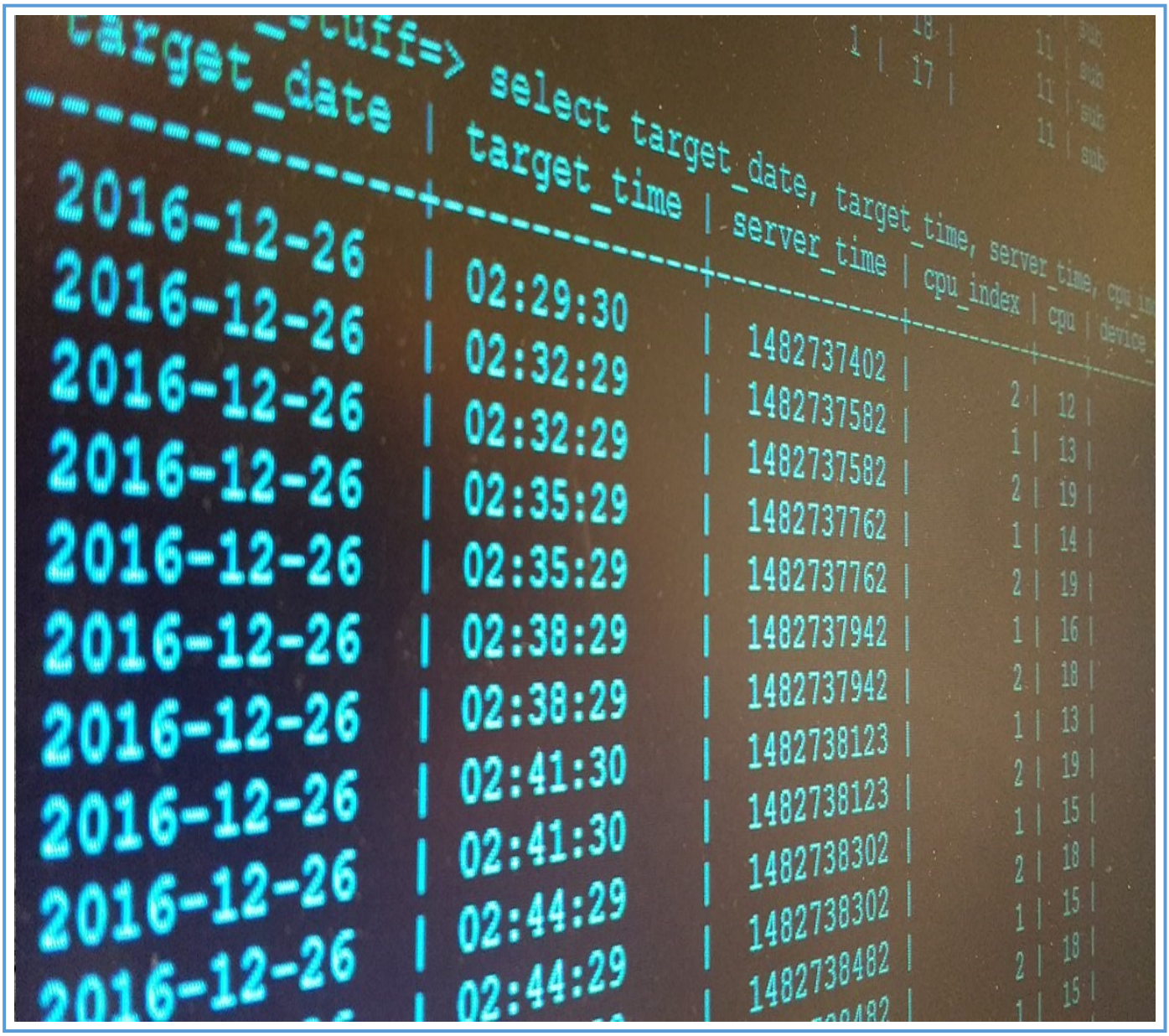


DELPHI SQL NOTES
www.mesoclever.com
Updated: April 2023

SQL COMMAND SUMMARY

SQL	Description	Usage
SELECT	Used to select data from a database	`SELECT * FROM tblTowns`
WHERE	Used to specify a filter	`SELECT * FROM tblTowns WHERE Population > 1000`
AS	Used to assign a name to a column	`SELECT MAX(Population) as HighestPopulation from tblTowns;`
LIKE	Used to search for a pattern	`SELECT * FROM tblTowns WHERE Province LIKE 'G%';`
IN	Used to specify multiple values	`SELECT * FROM tblTowns WHERE Province IN ('Gauteng','Eastern Cape') ;`
BETWEEN	Used to select a given range	`SELECT * FROM tblTowns WHERE Population BETWEEN`
ORDER BY	Used to sort data in ascending or descending	`SELECT * FROM tblTowns ORDER BY TownName;`
GROUP BY	Used in conjuntion with Aggregate Functions to group data	`SELECT Province, COUNT (*) AS CriticalTowns FROM`
		`tblTowns WHERE WaterRe-strictions = TRUE GROUP BY Province;`
INSERT INTO	Inserts data from a table	`INSERT INTO tblGames VALUES (76, #2017/12/24#, "HM008", 250, 5566);`
UPDATE	Updates data from in a table	`UPDATE tblTowns SET Wa-terRestrictions = True WHERE Province = "North West";`
DELETE	Delete data from a table	`DELETE FROM tblDams WHERE HeightOfWall <`
COUNT	Return the number of occurences	`SELECT Count(Population) FROM tblTowns;`
MIN	Returns smallest value of a column	`SELECT MIN(Population) from tblTowns;`
MAX	Returns highest value of a column	`SELECT MAX(Population) from tblTowns;`
AVG	Returns the average of a column	`SELECT AVG(Population) from tblTowns;`
SUM	Returns the total of a column	`SELECT SUM(Population) from tblTowns;`
NOW	Returns current date and time	`SELECT NOW();`
DAY	Returns day of month for a given date	`SELECT DAY(#2019-01-23#);`
MONTH	Returns the month part of a date	`SELECT MONTH(#2011-02-14#);`
YEAR	Returns the year part of a date	`SELECT YEAR(#2015-04-14#);`
ROUND	Rounds a number to a specific number of decimal places	`SELECT ROUND(123.321, 2);`
FORMAT	Rounds a number to a specific number of decimal places, returns string	`SELECT FORMAT (48934892.34893, 3);`

Welcome

Hello,

These notes cover 8 SQL questions, that appeared in past year papers. Please make sure to download the corresponding working files.

Downloads the Working Files

- Dams
- Dance
- Hockey
- Lan
- Payments
- Video
- Zoo
- Shop

The files can be downloaded from the below link:

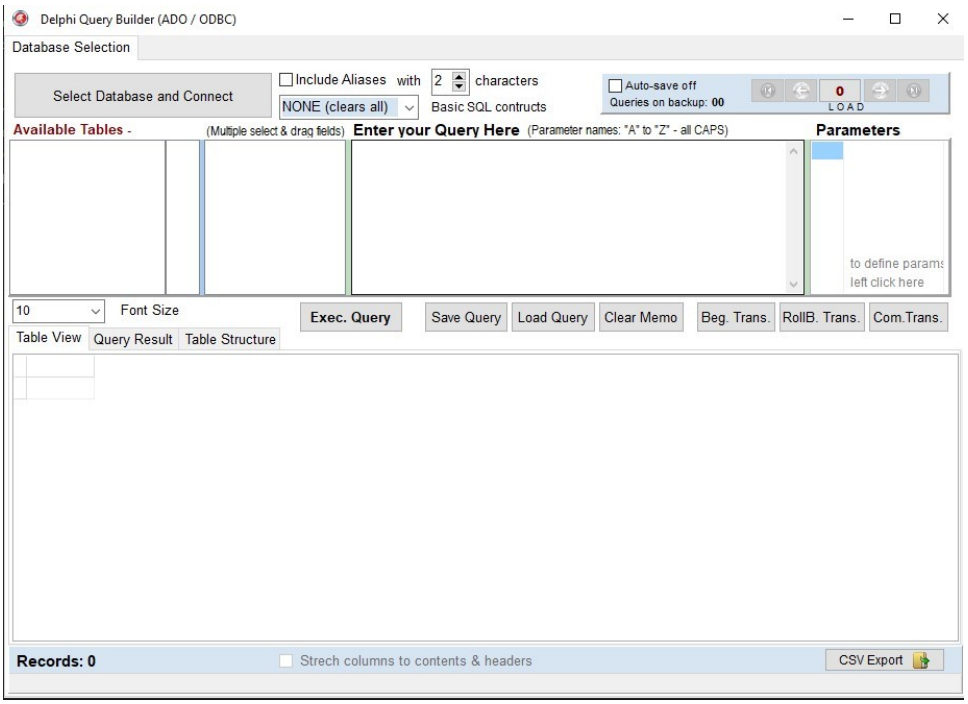
https://github.com/mesoclever/delphi/tree/master/SQL

Table Of Contents

Inside this issue:

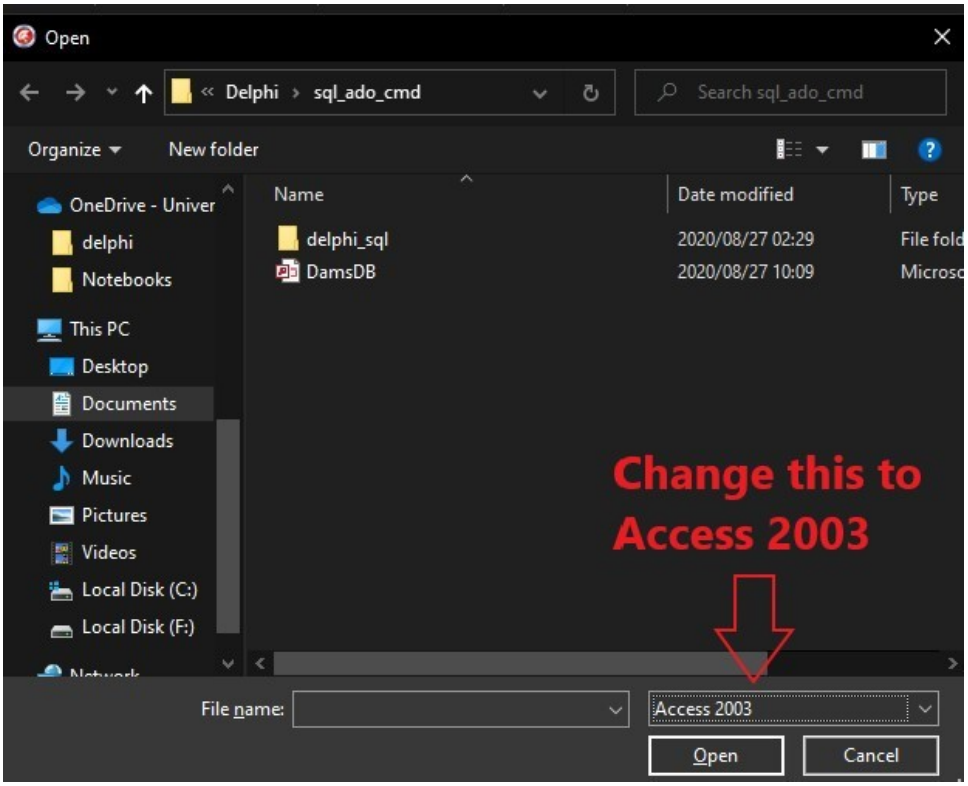
Delphi Query Builder

To practice these commands seen in this document we will make use of DelphiQryBuilder You must have downloaded the files first before you continue. See the 3rd page. (Found in the Working Files)



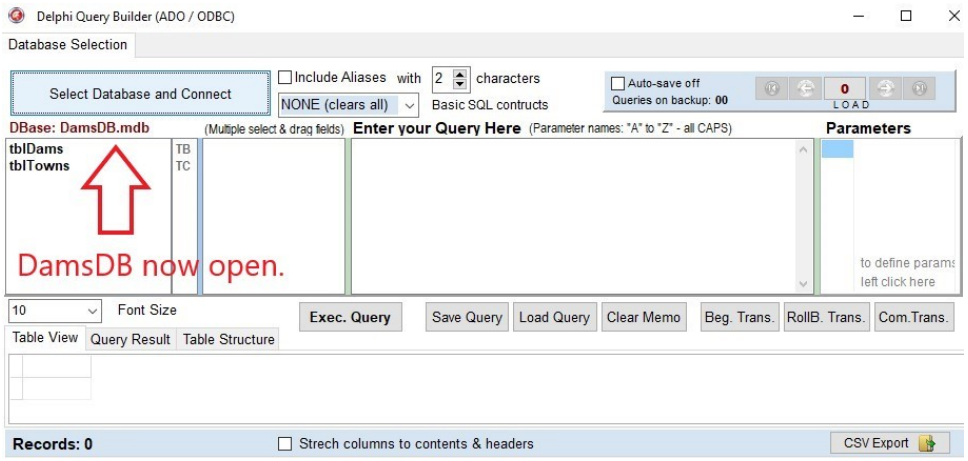
Opening a Database

1. To open a .mdb database, select **Select Database and Connect** in the top left corner.
2. Navigate to the folder where the .mdb file is located, and change the filetype to **Access 2003** . Your .mdb data base will then become visible. Select the database, and choose open.

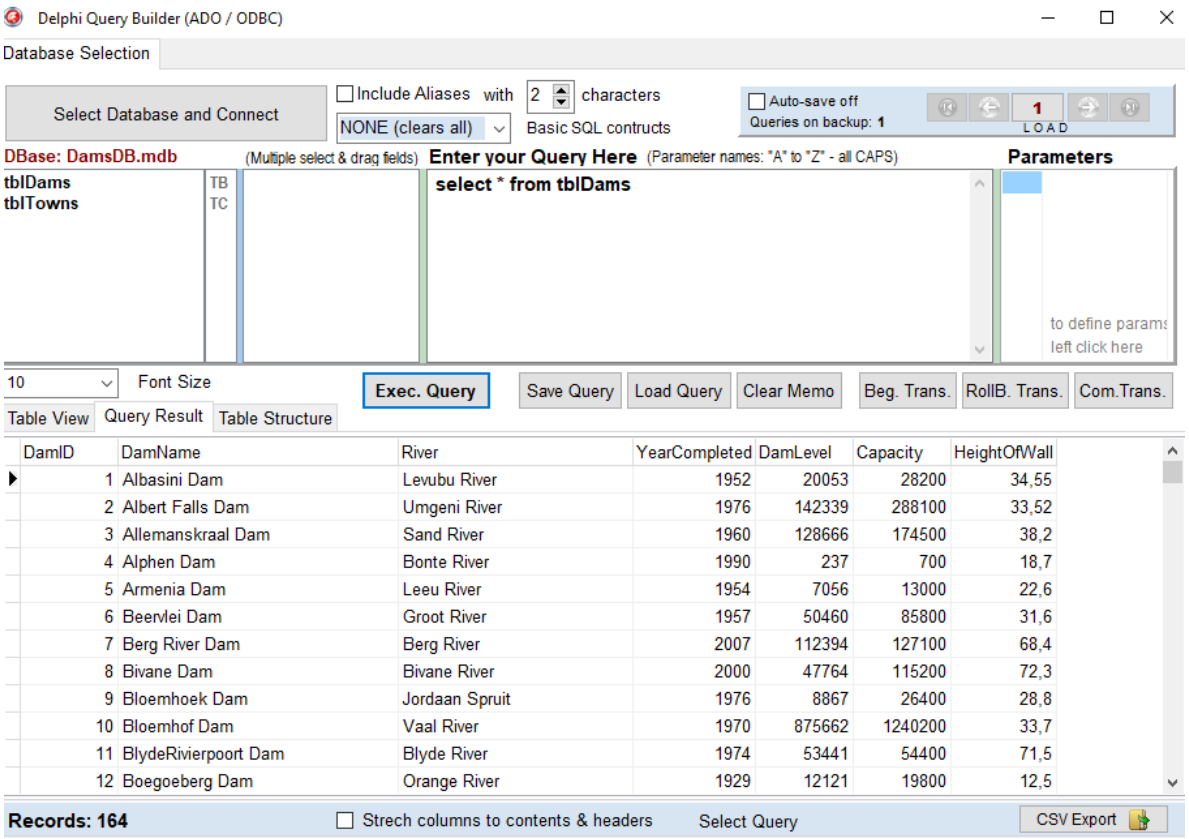


Your .mdb data base will then become visible. Select the database, and choose open.

3. Your .mdb data base will then become visible. Select the database, and choose open. Upon successfully opening the database, you should see the tables now present in the program:



To run a query insert the SQL query inside of “Enter you Query Here”, and click **Exec. Query**



SQL QUESTION COMMAND SUMMARY

```
// DamsDB
- select * from tblDams order by HeightOfWall;
- select TownName, Population from tblTowns where Population > 10000 and Province =
"Gauteng";
- select DamID, DamName, (Year(Now()) - YearCompleted) as Age, Round(DamLevel /
Capacity * 100, 1) as Percentage from tblDams;
- select Province, Count(*) as CriticalTowns from tblTowns where WaterRestrictions =
TRUE group by Province;
- select distinct Province from tblTowns, tblDams where tblTowns.DamID =
tblDams.DamID and River = "Vaal River";
- update tblTowns set WaterRestrictions = True where Province = "North West";
- delete from tblDams where HeightOfWall < 11.50;
```

```
// DanceDB
- select * from tblResults order by TypeOfDance;
- select RoutineNo, Week, TypeOfDance, Score from tblResults where (Score between 25
and 35) and (Week=5 or Week=9);
- select TypeOfDance, Count(*) as NumberOfPerformances from tblResults where
TypeOfDance="Waltz" group by TypeOfDance;
- select song, DancePartner1, DancePartner2 from tblDanceCouples, tblResults where
tblResults.DanceCoupleID=tblDanceCouples.DanceCoupleID and (ProfessionalDancers="B")
and ((Song like "Love%") OR (Song like "%you%"));
- select DanceCoupleID, Format((Sum(Score)/Count(*)),"0.000") as AverageScore from
tblResults group by DanceCoupleID;
- select distinct DancePartner1, DancePartner2 from tblResults, tblDanceCouples where
(tblResults.DanceCoupleID=tblDanceCouples.DanceCoupleID) and (Result like
"Eliminated") and (week < 12);
- update tblResults set Result="WINNERS" where Round = 2 and DanceCoupleID = 8;
```

```
// HockeyDB
- select PlayerSurname, PlayerName from tblPlayers where SkillsLevel=10;
- select Coach, TeamName from tblTeams where TeamName like "%B";
- select TeamName, Coach, (NumberOfGamesWon/NumberofGamesPlayed*100) as
PercentageGamesWon from tblTeams where TeamName = "u/14 A";
- select TeamName, Round(Avg(SkillsLevel),1) as AverageSkillsLevel from tblPlayers
group by TeamName having Avg(SkillsLevel) > 6;
- update tblTeams set NumberOfGamesWon = NumberOfGamesWon + 1 where TeamName <> "u/14
B";
```

```
// LanDB
- select * from tblPlayers order by Name;
- select Name, DateOfBirth from tblPlayers where Month(DateOfBirth) = 9;
- select GameDate,Round(Avg(Duration),1) As AverageDuration from tblGames group by
GameDate;
- select Name, Max(Score) as HighestScore from tblPlayers P, tblGames G where
P.PlayerID = G.PlayerID group by Name;
- insert into tblGames values (76, #2017/12/24#, "HM008", 250, 5566);
```

```
// PaymentsDB
- select * from tblEmployees order by Surname Asc;
- select Surname, FirstName, Children from tblEmployees where Children > 3 and
Permanent = TRUE;
- select PaymentNumber, IDNumber from tblEmployees E, tblPayments P where
E.EmployeeNumber = P.EmployeeNumber and PaymentDate = #2017/01/17#;
- delete from tblPlayers where PaymentNumber = 110;
- select Month(PaymentDate) as MonthNum format(Sum(GrossSalary-
Deductions),"Currency") as TotalAmountPaid from tblPayments group by Month
(PaymentDate);
```

SQL QUESTION COMMAND SUMMARY

```
// VideoDB
- select DateCreated, ProfileName from Profile order by DateCreated Desc;
- select DatePublished, format((Likes-Dislikes+Comments)/Views,"Currency",2) as
Earnings from Video, Profile where Video.ProfileID = Profile.ProfileID and Profile-
Name="DonDoe";
- select VideoTitle from Video where Views >= 1000 and Dislikes <= 10;
- Update Profile set ProfileName = "Tim Horton" where ProfileName = "TimmyTom";
```

```
// ZooDB
- select * from tblCarnivores order by FamilyName, ScientificName;
- select ScientificName, GeneralName, EnclosureNo, EnclosureSize from tblCarnivores
where EnclosureNo Like "ZE%" and FamilyName = "Herpestidae";
- select Endangered, Count(*) as CountAnimals from tblCarnivores Group By Endangered;
- select EnclosureNo, Format(EnclosureSize / (NumAdults+NumYoung),"0.00") as
SpacePerAnimal from tblCarnivores where GeneralName Like "%mongoose";
- update tblCarnivores set NumYoung = NumYoung + 3 where EnclosureNo="ZF1";
```

```
// ShopDB
-  sSQL1 := 'select StoreName from tblShops where StoreType = "Fashion" and Floor
="Upper"';
- sSQL2 := 'Select distinct StoreType from tblShops';
- sSQL3 := 'Select storeName, storeType, floor from tblShops where RiskFactor =
"'+sLevel+'" ';
- sSQL4 := 'Select StoreType, format(Sum(RentAmount) ,"Currency")  as [TotalRental]
from tblShops, tblRental where tblShops.shopID= tblRental.ShopID group by StoreType
having sum(RentAmount) > 100000;';
- sSQL5 := 'Update tblShops set ManagerGender = "Male" where ManagerGender Like
"Polygender" or Managergender like "Bigender" ';
```

```
// TrialsDB
- sSQL1 := 'select StoreName from tblShops where StoreType = "Fashion" and Floor
="Upper"';
- sSQL2 := 'Select distinct StoreType from tblShops';
- sSQL3 := 'Select storeName, storeType, floor from tblShops where RiskFactor =
"'+sLevel+'" ';
- sSQL4 := 'Select StoreType, format(Sum(RentAmount) ,"Currency")  as [TotalRental]
from tblShops, tblRental where tblShops.shopID= tblRental.ShopID group by StoreType
having sum(RentAmount) > 100000;';
- sSQL5 := 'Update tblShops set ManagerGender = "Male" where ManagerGender Like
"Polygender" or Managergender like "Bigender" ';
```

```
// Adding User Variable
- 'select TeamName, Coach, (NumberOfGamesWon/NumberofGamesPlayed*100) as
PercentageGamesWon from tblTeams where TeamName = "' + sTeam + '"';
- 'select TypeOfDance, Count(*) as NumberOfPerformances from tblResults where
TypeOfDance="' + sx + '" group by TypeOfDance;';
- 'select DatePublished, format((Likes-Dislikes+Comments)/Views,"Currency",2) as
Earnings from Video, Profile where Video.ProfileID = Profile.ProfileID and
ProfileName="'+sLine+'"';
- 'select ScientificName, GeneralName, EnclosureNo, EnclosureSize from tblCarnivores
where EnclosureNo Like "ZE%" and FamilyName = "'+sx+'"';
```

SQL AGGREGATE, AS FUNCTIONS

Aggregate Functions

An aggregate function performs some mathematic operations on a column.

Usage :

```
SELECT AggregateFunction(ColumnName) FROM tableName;
```

Min : Returns the minimum value from a column:

```
SELECT MIN(Score) from tblResults;
```

Max : Return the maximum value from a column:

```
SELECT MAX(Score) from tblResults;
```

Sum : Returns the sum from a column :

```
SELECT SUM(Score) from tblResults;
```

Avg : Returns the average from a column :

```
SELECT AVG(Score) from tblResults;
```

Count : Counts the Number of values in a column :

```
SELECT Count(Song) from tblResults;
```

When using Aggregate functions on more than one column, you must use the Group By Statement :

Find how many times the "Cha-Cha" was danced per week :

```
SELECT Week,Count(TypeOfDance) from tblResults where
TypeOfDance='Cha-Cha' GROUP BY Week;
```

Find how many couples where Eliminated per week:

```
SELECT Week,Count(Result) from tblResults WHERE Result="Eliminated" GROUP BY
Week;
```

Find which TypeOfDance yielded the highest score :

```
SELECT TypeOfDance,AVG(Score) FROM tblResults GROUP BY TypeOfDance ORDER BY
AVG(Score) desc;
```

Display the names and average skill levels of all teams with an average skill level of more than 6 :

```
SELECT TeamName, Round(Avg(SkillsLevel),1) AS AverageSkillsLevel FROM
tblPlayers GROUP BY TeamName HAVING Avg(SkillsLevel) > 6;
```

As

The as statement renames a column in the output :

Usage:

```
SELECT column AS renameHere FROM table;
```

Renaming the output Column:

```
SELECT Song AS renameSong  from tblResults ;
```

SQL BETWEEN, DATE

Between

The between statement selects values in a certain range.

Usage:

```
SELECT column1 FROM table BETWEEN value1 and value2
```

Show those DanceCoupleIDs that have a score in the range of 20 and 30:

```
SELECT DanceCoupleID,Score FROM tblResults WHERE Score BETWEEN 20 AND 30;
```

Which could have also been rewritten as:

```
SELECT DanceCoupleID,Score FROM tblResults WHERE Score >= 20 AND Score <= 30;
```

Show those Videos that where published beteween the dates 23 January 2015 and 31 December 2015:

```
SELECT VideoTitle, DatePublished FROM Video WHERE DatePublished BETWEEN
#2015/01/23# AND #2015/12/31#
```

Dates

You can make use of Date in the following ways:

Select a Date from the table based on some criteria:

```
SELECT GameDate FROM tblGames WHERE GameDate=#2016/01/11#
```

Select the current Date using NOW():

```
SELECT NOW();
```

Select the Day from a Date using DAY():

```
SELECT DAY(#2016/01/11#);
```

Select the Month from a Date using MONTH():

```
SELECT MONTH(#2016/01/11#);
```

Select the Year from a Date using YEAR():

```
SELECT YEAR(#2016/01/11#);
```

Select those Players born in September:

```
SELECT Name, DateOfBirth FROM tblPlayers where Month(DateOfBirth) = 9;
```

Show the age of each Dam:

```
SELECT DamName, (Year(Now()) - YearCompleted) as Age from tblDams;
```

Insert a New Date into the table:

```
INSERT INTO tblGames VALUES (76, #2017/12/24#, "HM008", 250, 5566);
```

SQL FORMAT, HAVING, DELETE, DISTINCT

Format

The Format statement, formats a number into a specific format :

Usage:

```
SELECT FORMAT(column1,1) FROM table;
```

Format a column to 2 Decimal Places:

```
SELECT FORMAT(Score,"0.00") from tblGames;
```

Working out the Average Score, and displaying to three decimal places

```
SELECT DanceCoupleID, FORMAT(AVG(Score),"0.000") AS AverageScore FROM
tblResults GROUP BY DanceCoupleID;
```

Using Format with "Currency":

```
SELECT MONTH(PaymentDate) as MonthNum FORMAT(Sum(GrossSalary-
Deductions),"Currency") AS TotalAmountPaid FROM tblPayments GROUP BY MONTH
(PaymentDate);
```

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 WHERE statement is applied before the GROUP BY. Another difference is that the HAVING STATEMENT can filter aggregrate results. The syntax for this statement is as follows:

Usage:

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

For every TypeOfDance, get the Max Score that is greater than 35:

```
SELECT TypeOfDance,MAX(Score) FROM tblResults GROUP BY TypeOfDance HAVING MAX
(Score) > 35;
```

Delete

The DELETE statement will delete a row within a table.

Usage:

```
DELETE FROM table WHERE condition;
```

Example:

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

Distinct

The Distinct statement eliminates duplicate values from a column, and only returns distinct values.

Usage:

```
SELECT DISTINCT column1, colum FROM table;
```

Show the different dance types:

```
SELECT DISTINCT TypeOfDance FROM tblResults;
```

Example:

```
SELECT DISTINCT Province FROM tblTowns, tblDams WHERE tblTowns.DamID =
tblDams.DamID AND River = "Vaal River";
```

SQL IN, GROUP BY, INSERT

In

The IN statement allows us to state which values we would like to see for a certain column.

Usage:

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

All couples that used the Cha-Cha and Salsa Dance :

```
SELECT DanceCoupleID,TypeOfDance FROM tblResults WHERE TypeOfDance
IN('Cha-Cha','Salsa');
```

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.

Usage:

```
SELECT columnsFROM table WHERE condition GROUP BY column.
```

Example:

```
SELECT Week,Count(TypeOfDance),TypeOfDance FROM tblResults WHERE
TypeOfDance='Cha-Cha' GROUP BY Week;

SELECT Endangered, Count(*) AS CountAnimals FROM tblCarnivores Group By
Endangered;

SELECT Month(PaymentDate) AS MonthNum FORMAT(Sum(GrossSalary-
Deductions),"Currency") AS TotalAmountPaid FROM tblPayments GROUP BY MONTH
(PaymentDate);
```

Insert

The INSERT statement allows us to add a row into a table.

Usage:

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

Example :

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

You can also use the INSERT statement like this :

Usage:

```
INSERT INTO table values(value1, value2);
```

Example:

```
INSERT INTO tblGames VALUES (76, #2017/12/24#, "HM008", 250, 5566);
```

SQL LIKE, ORDER BY, SELECT

Like

The Like operator allows us to search for a matching pattern within a column.

Usage:

```
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
```

Find all Songs that start with B:

```
SELECT Song from tblResults where SONG LIKE('B%');
```

Find all the songs that contain the word 'Love' within them:

```
SELECT Song from tblResults where SONG LIKE('%LOVE%');
```

Find all RoutineNos that obtained a score of 30 or greater:

```
  SELECT RoutineNo,Score from tblResults WHERE Score LIKE('3_');
```

Order By

The order by statement is used to arrange output in alphabetical order. When the order by statement is not specified, results from the table are displayed in the order they are found. With order by you can specify Ascending or Descending order:

Usage:

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

Example:

```
SELECT DanceCoupleID,TypeOfDance from tblResults where result =
"Safe" ORDER BY TypeOfDance desc;
```

Select

The select statement retrieves data from the table, based on the column you specify.

Usage:

```
SELECT column FROM table;
```

Select Single Column from a Table:

```
SELECT Song FROM tblResults ;
```

Select Specific Column :

```
SELECT TypeOfDance,Song,Score FROM tblResults;
```

Select All Columns from a Table :

```
SELECT * FROM tblResults ;
```

SQL UPDATE, WHERE

Update

The UPDATE statement allows us to update a field value within a table.

Usage:

```
UPDATE table SET column WHERE condition;
```

Example:

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

Where

The Where clause allows us to specify criteria, to narrow down our results to what we want.

Usage:

```
SELECT column FROM table WHERE criteria;
```

SQL IS NULL, NOT NULL

 # IS NULL

Selects all records that have no value for a specific field.

Usage:

WHERE field_name IS NULL

Example:

SELECT * FROM tblMovies WHERE genre IS NULL;

 # IS NOT NULL

Selects all records that have value for a specific field.

Usage:

WHERE field_name IS NOT NULL;

Example:

select deliveryaddress, (deliverydate -orderdate) as DaysToDeliver from orders where deliverydate is not null;