Create a Database

CREATE DATABASE <DB\_name>;

Use Database

USE <DB\_name>;

Create Table

CREATE TABLE <table\_name>

Delete Table

DROP TABLE <table\_name>

Different types of data

* CHAR() – Set length of data and it always has to be that length
* VARCHAR() – Set length of data and entries can be any length up to that
* INT - integer
* DECIMAL/NUMERIC – Number with a decimal point
* BINARY – string of 0 & 1’s (up to 16 bits)
* FLOAT – large mathematical number e.g. pi
* BIT – 0 or 1, boolean

Create Table

USE my\_db;

CREATE table film\_table

(

film\_name VARCHAR(50) ,

film\_type VARCHAR(25),

date\_of\_release DATE,

Director VARCHAR(30),

Writer VARCHAR(30),

Star\_Rating DECIMAL(2,1),

Film\_Language VARCHAR(20),

Official\_Website VARCHAR(50),

Plot\_Summary VARCHAR(MAX);

)

Alter Table

USE my\_db;

ALTER table film\_table ADD Trailer\_link VARCHAR(128);

Modify Table

USE my\_db;

ALTER TABLE film\_table

ALTER COLUMN Trailer\_link VARCHAR(64);

Change Data in Table

UPDATE <table\_name>

SET <column\_name\_for \_change>= ‘<NEW DATA INPUT>

WHERE <column\_name\_condition> = <condition>

UPDATE film\_table

SET Plot\_Summary = 'An overachieving London police officer is relocated to a country town with a dark secret'

WHERE Official\_Website = 'google.com';

Delete all data in a table

DELETE FROM <table\_name>

DELETE FROM film\_table;

Delete data from a row

DELETE FROM <table\_name>

WHERE <column\_name\_condition> = <condition> #try and use a unique data point for deletion

DELETE FROM film\_table

WHERE Trailer\_link IS NULL;

DELETE FROM film\_table

WHERE film\_name = 'Hot Fuzz';

Using NOT NULL

CREATE TABLE film\_table

(

Film\_name VARCHAR(10) NOT NULL,

Film\_type VARCHAR(25) NOT NULL,

Film\_cost DEC(3,2) NOT NULL DEFAULT 0

)

Making a Primary Key when making the table

CREATE TABLE film\_table

(

Film\_name VARCHAR(10) NOT NULL IDENTITY PRIMARY KEY,

Film\_type VARCHAR(25) NOT NULL,

Film\_cost DEC(3,2) NOT NULL DEFAULT 0

)

*Can ONLY make a primary key when making the table. It is immutable and cannot be changed after the table is made. If you don’t set it when you make the table, you’ll have to delete the whole table and make a new one to have a primary key.*

Database considerations

* Data Security
* Data Recovery
* Data Integrity
* Normal Form

Normal Form

A database is in its **First Normal Form** when the following conditions are satisfied:

* Make everything atomic – as small as it can be
* There should be no repeating groups

A database is in **Second Normal Form** when the following conditions are satisfied:

* It is in *First Normal Form*
* All non-key attributes are fully functional dependent on the *Primary Key*

A database is in it **Third Normal Form** when the following conditions are satisfied:

* It is in *Second Normal Form*
* There is no *transitive functional dependency*: i.e. when a non-key column is functionally dependent on another non-key column, which is functionally dependent on the primary key

Using SQL as a Tester

* Getting data for testing
* Saving data, generated during testing activity
* Data verifications in databases
  + Find data
  + Ensure data integrity
  + Manipulate test data for specific tests
* Testing databases

SQL SELECT SYNTAX

* SELECT
* DISTINCT
* FROM
* WHERE
* GROUP BY
* HAVING
* ORDER BY

Wildcards

% - substitute for zero or more characters

\_ - substitute for a single character

[character \_list] – brings back anything starting with those letters

[^character\_list] – brings back anything that does not start with those letters.

String Function

|  |  |
| --- | --- |
| SUBSTRING | SUBSTRING(expression, start, length) – makes a substring of an existing string |
| CHARINDEX | CHARINDEX(‘a’,’text’) – “find ‘a’ in column called ‘text’” |
| LEFT or RIGHT | LEFT(name, 5) for the first or last 5 characters – take the left/right most part of a string |
| LTRIM or RTRIM | Used to remove spaces at the beginning or end of a string |
| LEN | LEN(name) - the length of the string |
| REPLACE | Replace things. e.g. Replace(name, ‘ ‘, ‘\_\_\_’) to replace spaces with underscore in the column |
| UPPER or LOWER | UPPER(name) to convert to all upper (or lower) case |

DATES

|  |  |
| --- | --- |
| GETDATE | SELECT GETDATE() – returns current date & time |
| SYSDATETIME | SELECT SYSDATETIME() – returns the date and time of the computer being used |
| DATEADD | DATEADD(d/m/Y,5,OrderDate) AS “Due Date” to add 5 days/months/Years |
| DATEDIFF | DATEDIFF(d, OrderDate, ShippedDate) to calculate the difference between two dates |
| YEAR | SELECT YEAR(OrderDate) AS “Order Year” gets year from a date |
| MONTH | SELECT MONTH(OrderDate) AS “Order Month” gets month from a date |
| DAY | SELECT DAY(OrderDate) AS “Order Day” gets day from a date |

SELECT CASE

SELECT CASE

WHEN DATEDIFF(d, OrderDate, ShippedDate) < 10 THEN ‘On Time’

ELSE ‘Overdue’

END AS ‘Status’

FROM Orders

GROUPING

Aggregate Functions:

|  |  |
| --- | --- |
| SUM | SUM(OrderTotal) – grand total of a column for all rows selected |
| AVG | AVG(UnitPrice) – average of the column for all rows selected |
| MIN | MIN(UnitPrice) – for the smallest value in a column for all rows |
| Max | MAX(UnitPrice) - for the largest value in a column for all rows |
| COUNT | COUNT(\*) – for the number of NOT NULL rows selected. If \* is used then all rows are counted, regardless of NULL/NOTNULL. |

These must be used in conjunction with a **GROUP BY** clause.

**HAVING** is essentially a WHERE condition used *only* for when a GROUP BY clause is in effect. It limits the data in the query based on a given condition.