Systems Design & Databases

SQL Server - TSQL Queries to support:

[Super heroes]

name: Aram Brunton

course: Software Engineering BSc

date: 21st March 2025

Tutor Name:

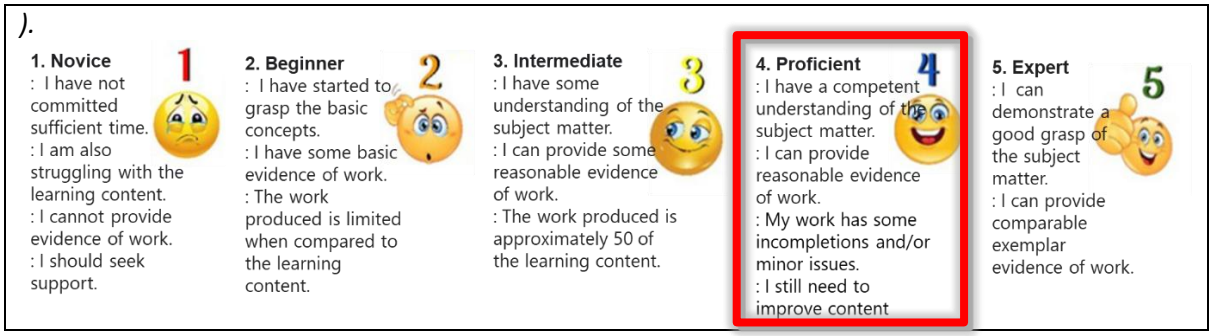
Table of Appendices

SQL Server Practitioner Details:

Please enter your details below:

SQL Server - TSQL Practitioner Details:		
	Name:	**Aram Brunton**
	Email Address:	E4054759@live.tees.ac.uk
	Course:	Software Engineering BSc
	Date:	21st March 2025
	Tutor:	

a) SQL Server Practitioner Performance Rating:



b) Introduction to the SQL Practitioner:

I decided upon studying to pursue my interests in becoming a graduate developer as when I was younger and used software and websites, I was always curious on how they worked and functioned. When I was in primary school, I taught myself HTML and CSS and felt a fire light inside of me when I made something that worked, as if it was a dopamine spike. This feeling of making something myself then made me progress into learning programming languages such as Python, C# and Java, which gave me a wider option of programs I could create, from game server scripting to mobile applications and websites. My aspirations as a junior graduate developer are to expand my skill set and to discover new creative ideas that can better society, I want to be an innovative developer who has a wide range of skills when it comes to computer science.

c) Why you should learn SQL:

SQL is a valuable skill to have under your belt when you are becoming a developer, not only for programming tasks but for data analysis in general. SQL can be used to store data of users, hold information relative to the task at hand, and to generally interact with a database model. Most technologies and software use SQL to collect and store relevant data to the database. Back-end developers should learn SQL as it will allow them to store large amounts of data that are necessary for the function of the project they are developing.

SQL Server Database Overview:

a) SQL Server Database for Demos:

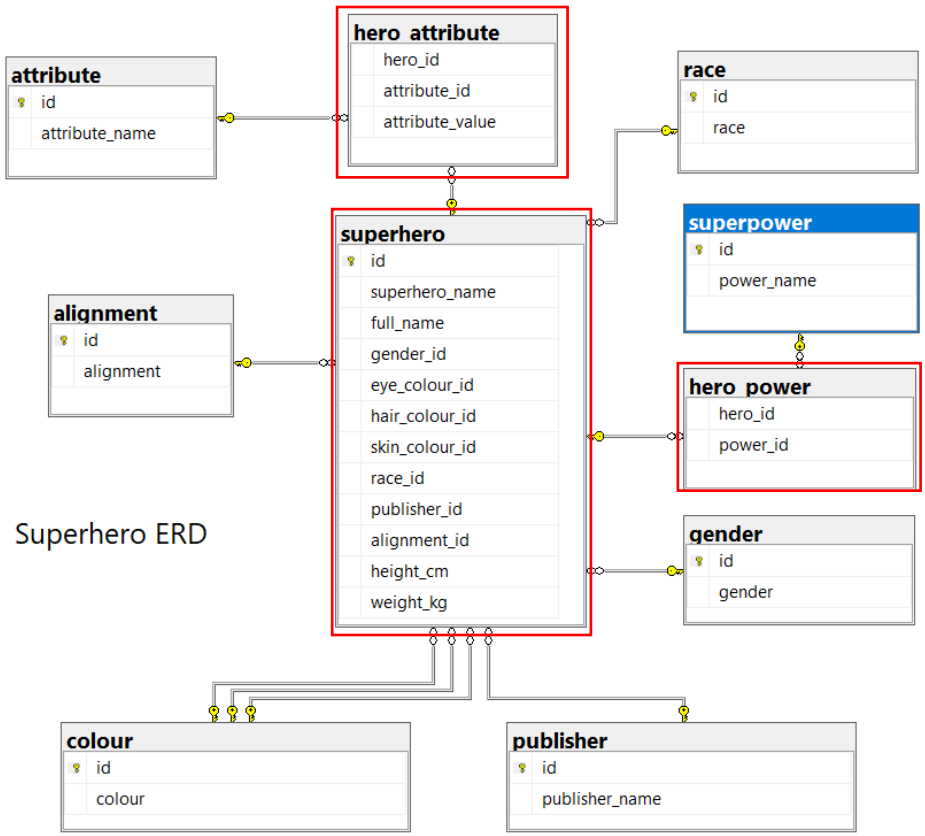
I am going to investigate the SQL Server Database **Superheroes** to develop range of useful TSQL Queries and Scripts to support **Action Game and Superhero Dictionary Website Developers.**

The aim is to provide useful patterns of data to serve front end development technologies such as Web or Mobile Applications.

Provided below in this document are examples of my best TSQL Demos (Queries and Scripts) to support users of **Superheroes**

b) SQL Server Database Diagrams:

Below is the supporting ERD diagram for the superheroes database and all tables highlighted in a red box are tables of interest when writing my own queries.



SQL Practitioners TSQL Demos:

Introduction:

Provided below is an audit trail of my best examples of TSQL querying skills to support 'business functions' or user requirements for the superheroes database

TSQL Part 1: SQL Server Coding Basics

1. TSQL03 to TSQL08: SQL Server Basics

This section covers the basics skills in using SSMS and scoping TSQL Queries either by code or by using the **Design Query in Editor** how to use Select statements to query data from table(s), Join across related tables, sort and filtering with Where, modifying data and using built in functions for Superheroes:

.sql File for TSQL03-08 Demos:	superheroes.sql

a) Module 3: Writing SELECT Queries with single Table

Why write Select queries?

The purpose of the SELECT statement is to query database tables, apply logical manipulation to the data, and result a result set.

Demo A1: Writing Simple SELECT Query

[Instructions: utilise the structure by presenting your TSQL Demo code and results as follows]

Selecting relevant information about a superhero from the 'superhero' table

```
USE superhero
GO

-- Use this query to select all the rows from the superhero table
SELECT * FROM superhero;
```

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90
2	2	A-Bomb	Richard Milhouse Jones	1	33	1	1	24	13	1	203	441
3	3	Abe Sapien	Abraham Sapien	1	7	1	7	33	3	1	191	65
4	4	Abin Sur	-	1	7	1	23	55	4	1	185	90
5	5	Abomination	Emil Blonsky	1	14	1	1	28	13	2	203	441
6	6	Abraxas	Abraxas	1	7	4	1	12	13	2	0	0
7	7	Absorbing Man	NULL	1	7	1	1	24	13	2	193	122
8	8	Adam Monroe	-	1	7	6	1	1	15	1	0	0
9	9	Adam Strange	Adam Strange	1	7	6	1	24	4	1	185	88
10	10	Agent 13	Sharon Carter	2	7	6	1	1	13	1	173	61
11	11	Agent Bob	Bob	1	9	9	1	24	13	1	178	81
12	12	Agent Zero	Christoph Nord	1	1	1	1	1	13	1	191	104
13	13	Air-Walker	Gabriel Lan	1	7	31	1	1	13	2	188	108
14	14	Ajax	NULL	1	9	4	1	13	13	2	193	90
15	15	Alan Scott	NULL	1	7	6	1	1	4	1	180	90
16	16	Alex Mercer	Alexander J. Mercer	1	1	1	1	24	25	2	0	0
17	17	Alex Woolsly	Alex Woolsly	1	1	1	1	1	15	1	0	0
18	18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth	1	7	4	1	24	4	1	178	72
19	19	Alien	Xenomorph	1	1	1	4	57	3	2	244	169

```sql
-- Use this query to select certain columns from the superhero table
SELECT superhero_name, full_name, height_cm, weight_kg FROM superhero
```

Results | Messages

	superhero_name	full_name	height_cm	weight_kg
1	3-D Man	Charles Chandler	188	90
2	A-Bomb	Richard Milhouse Jones	203	441
3	Abe Sapien	Abraham Sapien	191	65
4	Abin Sur	-	185	90
5	Abomination	Emil Blonsky	203	441
6	Abraxas	Abraxas	0	0
7	Absorbing Man	NULL	193	122
8	Adam Monroe	-	0	0
9	Adam Strange	Adam Strange	185	88
10	Agent 13	Sharon Carter	173	61
11	Agent Bob	Bob	178	81
12	Agent Zero	Christoph Nord	191	104
13	Air-Walker	Gabriel Lan	188	108
14	Ajax	NULL	193	90
15	Alan Scott	NULL	180	90
16	Alex Mercer	Alexander J. Mercer	0	0
17	Alex Woolsly	Alex Woolsly	0	0
18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth	178	72
19	Alien	Xenomorph	244	169

Demo A2: Eliminating Duplicates with DISTINCT

Selecting height for a superhero without any duplicates

```sql
USE superhero
GO

-- Use this query to display each recorded height of each superhero without any
duplicates
-- from the superhero table
SELECT DISTINCT height_cm FROM superhero;
```

100 %

Results | Messages

	height_cm
1	NULL
2	0
3	61
4	64
5	66
6	71
7	79
8	122
9	137
10	140
11	142
12	155
13	157
14	160
15	163
16	165
17	168
18	170
19	173

Demo A3: Using Column and Table Aliases Lesson

Selecting columns from the superhero table with aliases

```
USE superhero
GO

-- Use this query to select the column 'superhero_name' as 'KnownAs' and the column
'full_name' as 'LegalName' from the table 'superhero'
-- This changes the column name and changes it to an alias as stated above
SELECT
superhero_name AS KnownAs, full_name AS LegalName
FROM superhero;
```

Results | Messages

	KnownAs	LegalName
1	3-D Man	Charles Chandler
2	A-Bomb	Richard Milhouse Jones
3	Abe Sapien	Abraham Sapien
4	Abin Sur	-
5	Abomination	Emil Blonsky
6	Abraxas	Abraxas
7	Absorbing Man	NULL
8	Adam Monroe	-
9	Adam Strange	Adam Strange
10	Agent 13	Sharon Carter
11	Agent Bob	Bob
12	Agent Zero	Christoph Nord
13	Air-Walker	Gabriel Lan
14	Ajax	NULL
15	Alan Scott	NULL
16	Alex Mercer	Alexander J. Mercer
17	Alex Woolsly	Alex Woolsly
18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth
19	Alien	Xenomorph

Demo A4: Writing Simple CASE Expressions

Selecting superhero names and legal names and putting a place holder for superheroes with no known legal name

```
-- Use this query to select the superhero name and their legal name from the
superhero table but where the superhero
-- does not have a legal name as it is set to NULL or a '-', set it as "No Legal
Name"
SELECT
superhero_name,
CASE
       WHEN full_name IS NULL THEN 'No Legal Name'
       WHEN full_name = '-' THEN 'No Legal Name'
       ELSE full_name
END AS full_name
FROM superhero;
```

Results | Messages

	superhero_name	full_name
1	3-D Man	Charles Chandler
2	A-Bomb	Richard Milhouse Jones
3	Abe Sapien	Abraham Sapien
4	Abin Sur	No Legal Name
5	Abomination	Emil Blonsky
6	Abraxas	Abraxas
7	Absorbing Man	No Legal Name
8	Adam Monroe	No Legal Name
9	Adam Strange	Adam Strange
10	Agent 13	Sharon Carter
11	Agent Bob	Bob
12	Agent Zero	Christoph Nord
13	Air-Walker	Gabriel Lan
14	Ajax	No Legal Name
15	Alan Scott	No Legal Name
16	Alex Mercer	Alexander J. Mercer
17	Alex Woolsly	Alex Woolsly
18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth
19	Alien	Xenomorph

Query executed successfully. | ARAMSPC (16.0 RTM)

b) Module 4: Joining and Querying Multiple Tables

Why use Joining and Querying Multiple Tables?

By making multiple queries and joining the data in code will make multiple requests to your database, one for each table you need data from. The advantage of using a join in the SQL query will reduce the number of connection made to just one. This is especially advantageous if your database server is on a separate machine.

Demo B1: How to provide data from 2 related tables with a Join

Joining the gender of the superhero from the gender table with the superhero table using the ID as the primary key

```sql
USE superhero;
GO

-- gender_id is the common key used to link these two tables
SELECT * FROM superhero
SELECT * FROM gender

-- This query links the two tables, superhero and gender to create a new table
-- that displays the superhero name, their legal name and the gender of the super
hero
SELECT s.superhero_name, s.full_name, g.gender
FROM dbo.superhero AS s JOIN dbo.gender AS g ON s.gender_id = g.id;
```

100 %

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90
2	2	A-Bomb	Richard Milhouse Jones	1	33	1	1	24	13	1	203	441
3	3	Abe Sapien	Abraham Sapien	1	7	1	7	33	3	1	191	65
4	4	Abin Sur	-	1	7	1	23	55	4	1	185	90
5	5	Abomination	Emil Blonsky	1	14	1	1	28	13	2	203	441
6	6	Abraxas	Abraxas	1	7	4	1	12	13	2	0	0
7	7	Absorbing Man	NULL	1	7	1	1	24	13	2	193	122
8	8	Adam Monroe	-	1	7	6	1	1	15	1	0	0

	id	gender
1	1	Male
2	2	Female
3	3	N/A

	superhero_name	full_name	gender
1	3-D Man	Charles Chandler	Male
2	A-Bomb	Richard Milhouse Jones	Male
3	Abe Sapien	Abraham Sapien	Male
4	Abin Sur	-	Male
5	Abomination	Emil Blonsky	Male

Query executed successfully.

Demo B2: How to Query with Inner Joins

A query to select all superheroes who have an alignment in the database

```
USE superhero;
GO

-- This query selects all superheroes who have an alignment matching between the
-- tables 'superhero' and 'alignment' It only shows matches and does not show
-- fields that have a null value for alignment
SELECT s.superhero_name, s.full_name, a.alignment
FROM dbo.superhero AS s INNER JOIN dbo.alignment AS a ON s.alignment_id = a.id;
```

Results | Messages

	superhero_name	full_name	alignment
28	Angela	-	Bad
29	Animal Man	Bernhard Baker	Good
30	Annihilus	Annihilus	Bad
31	Ant-Man	Henry Jonathan Pym	Good
32	Ant-Man II	Scott Lang	Good
33	Anti-Monitor	NULL	Bad
34	Anti-Spawn	Jason Wynn	Bad
35	Apocalypse	En Sabah Nur	Bad
36	Aquababy	Arthur Curry, Jr.	Good
37	Aqualad	Garth	Good
38	Aquaman	Orin	Good
39	Arachne	Julia Carpenter	Good
40	Archangel	Warren Kenneth Worthington III	Good
41	Arclight	Philippa Sontag	Bad

Demo B3: How to Query with Outer Joins

A query that shows all the superhero alignments whether they have a match in the alignment table or not

```
USE superhero;
GO

-- This query will join the two tables 'superhero' and 'alignment' and will show
-- if a superhero does not have an allignment by having NULL as the value in the
-- alignment column
SELECT s.superhero_name, s.full_name, a.alignment
FROM dbo.superhero AS s FULL OUTER JOIN dbo.alignment AS a ON s.alignment_id = a.id;
```

	superhero_name	full_name	alignment
25	Angel	Liam	Good
26	Angel Dust	Christina	Good
27	Angel Salvadore	Angel Salvadore Boh...	Good
28	Angela	-	Bad
29	Animal Man	Bernhard Baker	Good
30	Annihilus	Annihilus	Bad
31	Ant-Man	Henry Jonathan Pym	Good
32	Ant-Man II	Scott Lang	Good
33	Anti-Monitor	NULL	Bad
34	Anti-Spawn	Jason Wynn	Bad
35	Anti-Venom	Edward Charles Allan ...	NULL
36	Apocalypse	En Sabah Nur	Bad
37	Aquababy	Arthur Curry, Jr.	Good
38	Aqualad	Garth	Good
39	Aquaman	Orin	Good
40	Arachne	Julia Carpenter	Good
41	Archangel	Warren Kenneth Wort...	Good
42	Arclight	Philippa Sontag	Bad
43	Ardina	-	Good
44	Ares	-	Neutral
45	Ares	-	Good
46	Ariel	Ariel	Good

Demo B4: How Query with Cross Joins and Self Joins

c) Module 5: Sorting and Filtering Data

Demo C1: How to Sort Data

Sorting Data by Ascending and Descending Order

```
-- Task: Sort Data by alphabetical order
-- Sorted data by alignment in alphabetical order
SELECT s.superhero_name, s.full_name, a.alignment
FROM dbo.superhero AS s INNER JOIN dbo.alignment AS a ON s.alignment_id = a.id
ORDER BY a.alignment;
```

Results | Messages

	superhero_name	full_name	alignment
190	T-X	Cyberdyne Syste...	Bad
191	Taskmaster	-	Bad
192	Thanos	Thanos	Bad
193	Thanos (Infinity...	-	Bad
194	Tiger Shark	Todd Arliss	Bad
195	Tinkerer	Phineas Mason	Bad
196	Trigon	-	Bad
197	Two-Face	Harvey Dent	Bad
198	Ultron	NULL	Bad
199	Utgard-Loki	-	Bad
200	Vanisher	-	Bad
201	Vegeta	-	Bad
202	Venom	Edward Charles Al...	Bad
203	Venom II	Angelo Fortunato	Bad
204	Venom III	MacDonald Gargan	Bad
205	Violator	-	Bad
206	Vulture	Adrian Toomes	Bad
207	Walrus	Hubert Carpent	Bad
208	Warp	Emil LaSalle	Bad
209	Weapon XI	Weapon XI	Bad
210	White Canary	-	Bad
211	Yellow Claw	NULL	Bad
212	Zoom	Hunter Zolomon	Bad
213	3-D Man	Charles Chandler	Good
214	A-Bomb	Richard Milhouse ...	Good
215	Abe Sapien	Abraham Sapien	Good
216	Abin Sur	-	Good
217	Adam Monroe	-	Good
218	Adam Strange	Adam Strange	Good
219	Agent 13	Sharon Carter	Good
220	Agent Bob	Bob	Good
221	Agent Zero	Christoph Nord	Good
222	Alan Scott	NULL	Good
223	Alex Woolsly	Alex Woolsly	Good

Query executed successfully.

```sql
-- Task: Sort Data by alphabetical order
-- Sorted data by alignment in alphabetical order descending
SELECT s.superhero_name, s.full_name, a.alignment
FROM dbo.superhero AS s INNER JOIN dbo.alignment AS a ON s.alignment_id = a.id
ORDER BY a.alignment DESC;
```

11	Gladiator	Kallark	Neutral
12	Indigo	-	Neutral
13	Juggernaut	Cain Marko	Neutral
14	Living Tribunal	-	Neutral
15	Lobo	-	Neutral
16	Lucifer Morningstar	-	Neutral
17	Man-Bat	Robert Kirkland Langstrom	Neutral
18	One-Above-All	-	Neutral
19	Phantom Stranger	-	Neutral
20	Raven	Rachel Roth	Neutral
21	Red Hood	Jason Peter Todd	Neutral
22	Red Hulk	Thaddeus E. Ross	Neutral
23	Robin VI	Carrie Kelley	Neutral
24	Sandman	-	Neutral
25	Sinestro	Thaal Sinestro	Neutral
26	The Comedian	Edward Morgen Blake	Neutral
27	The Presence	-	Neutral
28	Toad	Mortimer Toynbee	Neutral
29	3-D Man	Charles Chandler	Good
30	A-Bomb	Richard Milhouse Jones	Good
31	Abe Sapien	Abraham Sapien	Good
32	Abin Sur	-	Good
33	Adam Monroe	-	Good
34	Adam Strange	Adam Strange	Good
35	Agent 13	Sharon Carter	Good
36	Agent Bob	Bob	Good

Demo C2: How to Filter Data with Predicates

Filtering data using predicates using the WHERE statement

```sql
-- This query uses a predicate to find all the superhero rows that has
-- a superhero name starting with the letter 'a'
SELECT *
FROM superhero
WHERE superhero_name LIKE 'a%';
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_
1	2	A-Bomb	Richard Milhouse Jones	1	33	1	1
2	3	Abe Sapien	Abraham Sapien	1	7	1	7
3	4	Abin Sur	-	1	7	1	23
4	5	Abomination	Emil Blonsky	1	14	1	1
5	6	Abraxas	Abraxas	1	7	4	1
6	7	Absorbing Man	NULL	1	7	1	1
7	8	Adam Monroe	-	1	7	6	1
8	9	Adam Strange	Adam Strange	1	7	6	1
9	10	Agent 13	Sharon Carter	2	7	6	1
10	11	Agent Bob	Bob	1	9	9	1
11	12	Agent Zero	Christoph Nord	1	1	1	1
12	13	Air-Walker	Gabriel Lan	1	7	31	1
13	14	Ajax	NULL	1	9	4	1
14	15	Alan Scott	NULL	1	7	6	1
15	16	Alex Mercer	Alexander J. Mercer	1	1	1	1
16	17	Alex Woolsly	Alex Woolsly	1	1	1	1
17	18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth	1	7	4	1
18	19	Alien	Xenomorph	1	1	1	4

```sql
--This query uses a predicate to find all the superhero rows that have a
--weight greater than 120kg
SELECT *
FROM superhero
WHERE weight_kg > 120;
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	2	A-Bomb	Richard Milhouse Jones	1	33	1	1	24	13	1	203	441
2	5	Abomination	Emil Blonsky	1	14	1	1	28	13	2	203	441
3	7	Absorbing Man	NULL	1	7	1	1	24	13	2	193	122
4	19	Alien	Xenomorph	1	1	1	4	57	3	2	244	169
5	21	Amazo	-	1	23	1	1	5	4	2	257	173
6	31	Ant-Man	Henry Jonathan Pym	1	7	6	1	24	13	1	211	122
7	35	Anti-Venom	Edward Charles Allan Brock	1	7	6	1	52	13	NULL	229	358
8	36	Apocalypse	En Sabah Nur	1	23	4	13	42	13	2	213	135
9	39	Aquaman	Orin	1	7	6	1	8	4	1	185	146
10	44	Ares	-	1	23	4	1	21	4	3	208	162
11	45	Ares	-	1	9	9	1	1	13	1	185	270
12	50	Atlas	-	1	7	9	1	21	4	2	198	126
13	52	Atlas	Erik Stephan Josten	1	7	9	1	21	4	2	198	126
14	64	Bane	-	1	1	1	1	24	4	2	203	180
15	76	Battlestar	Lemar Hoskins	1	9	4	1	1	13	1	198	133
16	79	Beast	Henry Philip McCoy	1	7	7	7	42	13	1	180	181
17	83	Beta Ray Bill	Beta Ray Bill (translation of his Korbinite name)	1	1	1	1	1	13	1	201	216
18	85	Bi-Beast	-	1	4	1	1	5	13	1	229	158
19	86	Big Barda	Barda Free	2	7	4	1	44	4	2	188	135

Demo C3: How to Filter Data with TOP and OFFSET-FETCH

Gathering the top 5 heaviest superheroes from the superhero table

```sql
-- This query selects the top 5 heaviest super heroes from the
-- superhero table and orders them as the first result being the
-- heaviest to the last result being the 5th heaviest
SELECT TOP 5 *
FROM superhero
ORDER BY weight_kg DESC;
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	293	Godzilla	-	3	1	1	1	NULL	1	2	10800	90000000
2	402	King Kong	-	1	33	4	1	6	1	1	3050	9000000
3	706	Utgard-Loki	-	1	7	31	1	20	13	2	1520	58000
4	262	Fin Fang Foom	-	1	23	1	14	35	13	1	975	18000
5	312	Groot	Groot	1	33	1	1	19	13	1	701	4000

```sql
-- This query offsets the first 5 rows of the results so that it
-- retrieves more realistic weights that can be used by larger
-- superheroes
SELECT *
FROM superhero
ORDER BY weight_kg DESC
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	357	Iron Monger	Obadiah Stane	1	7	1	1	1	13	2	0	2000
2	593	Sasquatch	Walter Langkowski	1	23	19	1	1	13	1	305	900
3	386	Juggernaut	Cain Marko	1	7	23	1	24	13	3	287	855
4	210	Darkseid	Uxas	1	23	1	13	44	4	2	267	817
5	290	Giganta	Doris Zuel	2	14	23	1	1	4	2	6250	630

Demo C4: How to work with Unknown Values

Finding NULL Values from the race column of the superhero table

```sql
-- This query will select all rows from the superhero table
-- where the super hero does not have a race assigned.
SELECT *
FROM superhero
WHERE race_id IS NULL;
```

100 %

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	180	Chewbacca	-	1	7	9	1	NULL	5	1	226	111
2	293	Godzilla	-	3	1	1	1	NULL	1	2	10800	90000000
3	552	Quake	Daisy Louise Johnson	2	9	4	1	NULL	13	1	163	52
4	594	Saturn Girl	Imra Ardeen-Ranzz	2	7	6	1	NULL	4	1	170	59

```sql
-- This query will select all rows from the superhero table
-- where the super heroes do have a race assigned to them
```

```sql
SELECT *
FROM superhero
WHERE race_id IS NOT NULL;
```

100 %

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90
2	2	A-Bomb	Richard Milhouse Jones	1	33	1	1	24	13	1	203	441
3	3	Abe Sapien	Abraham Sapien	1	7	1	7	33	3	1	191	65
4	4	Abin Sur	-	1	7	1	23	55	4	1	185	90
5	5	Abomination	Emil Blonsky	1	14	1	1	28	13	2	203	441
6	6	Abraxas	Abraxas	1	7	4	1	12	13	2	0	0
7	7	Absorbing Man	NULL	1	7	1	1	24	13	2	193	122
8	8	Adam Monroe	-	1	7	6	1	1	15	1	0	0
9	9	Adam Strange	Adam Strange	1	7	6	1	24	4	1	185	88
10	10	Agent 13	Sharon Carter	2	7	6	1	1	13	1	173	61
11	11	Agent Bob	Bob	1	9	9	1	24	13	1	178	81
12	12	Agent Zero	Christoph Nord	1	1	1	1	1	13	1	191	104
13	13	Air-Walker	Gabriel Lan	1	7	31	1	1	13	2	188	108
14	14	Ajax	NULL	1	9	4	1	13	13	2	193	90
15	15	Alan Scott	NULL	1	7	6	1	1	4	1	180	90
16	16	Alex Mercer	Alexander J. Mercer	1	1	1	1	24	25	2	0	0
17	17	Alex Woolsly	Alex Woolsly	1	1	1	1	1	15	1	0	0
18	18	Alfred Pennyworth	Alfred Thaddeus Crane Pennyworth	1	7	4	1	24	4	1	178	72
19	19	Alien	Xenomorph	1	1	1	4	57	3	2	244	169

d) Module 6: Working with Data Types

Demo D1: Working with Data Type examples

Converting the height and weight to float values instead of integer values

```sql
-- This query will convert any date string inputted in the
-- second argument to the datatime data type
SELECT CONVERT(datetime, '01/05/2025');
```

Results | Messages

	(No column name)
1	2025-01-05 00:00:00.000

```sql
-- This query will change the inputted number as a float
-- to a string
SELECT CONVERT(varchar, 62.3) AS [Converted Data];
```

100 %

Results | Messages

	Converted Data
1	62.3

Demo D2: Working with Character Data

Using collate to filter inputs by case sensitivity

```sql
-- These two queries are case sensitive so the collate will only
-- return results of what matches the input ('3-D Man') if it is
-- Spelt with the same capitals as what it stored in the database
SELECT *
FROM superhero
WHERE superhero_name COLLATE Latin1_General_CS_AS = '3-D Man'
```

```sql
SELECT *
FROM superhero
WHERE superhero_name COLLATE Latin1_General_CS_AS = '3-D man'

-- These two queries are not case sensitive so the SELECT will
-- return the row whether or not the input has the correct
-- capital letters.
SELECT *
FROM superhero
WHERE superhero_name COLLATE Latin1_General_CI_AS = '3-D Man'

SELECT *
FROM superhero
WHERE superhero_name COLLATE Latin1_General_CI_AS = '3-D man'
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90

id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	1	3-D Man	Charles Chandler	1	9	13	1	1	13	1	188	90

Demo D3: Working with Date and Time Data

These queries show how you can use the date and time data to find specific fields of data.

```sql
-- THis query will select the movie released on the date specified
SELECT *
FROM movie_releases
WHERE movie_release = '2008-07-18'

-- This query will return all movies that were released in 2008
-- from the movie_releases table
SELECT *
FROM movie_releases
WHERE YEAR(movie_release) = 2008;

-- This query will show me all the movies that were released
-- after 2009
SELECT *
FROM movie_releases
WHERE YEAR(movie_release) > 2009

-- This query will show me all movies that have been released
-- between 2012 and 2018 from the movie_releases table
SELECT *
FROM movie_releases
WHERE YEAR(movie_release) BETWEEN '2012' AND '2018';
```

100 %

Results | Messages

	movie_id	movie_name	movie_publisher	movie_release
1	2	The Dark Knight	4	2008-07-18 00:00:00.000

	movie_id	movie_name	movie_publisher	movie_release
1	1	Iron Man	13	2008-05-02 00:00:00.000
2	2	The Dark Knight	4	2008-07-18 00:00:00.000

	movie_id	movie_name	movie_publisher	movie_release
1	3	Captain America: The First Avenger	13	2011-07-22 00:00:00.000
2	4	Man of Steel	4	2013-06-14 00:00:00.000
3	5	Black Panther	13	2018-02-16 00:00:00.000
4	6	Wonder Woman	4	2017-06-02 00:00:00.000
5	7	Spider-Man: Homecoming	13	2017-07-07 00:00:00.000
6	8	The Avengers	13	2012-05-04 00:00:00.000
7	9	Justice League	4	2017-11-17 00:00:00.000
8	10	Doctor Strange	13	2016-11-04 00:00:00.000

	movie_id	movie_name	movie_publisher	movie_release
1	4	Man of Steel	4	2013-06-14 00:00:00.000
2	5	Black Panther	13	2018-02-16 00:00:00.000
3	6	Wonder Wo...	4	2017-06-02 00:00:00.000
4	7	Spider-Man:...	13	2017-07-07 00:00:00.000

Query executed successfully. | ARAMSPC (16.0 RTM) | ARAMSPC\amdru (69) | superhero | 00:00:00 | 26 rows

e) Module 7: Using DML to Modify Data

Why use Using DML to Modify Data?

DML is an abbreviation for Data Manipulation Language. Represents a collection of programming languages explicitly used to make changes to the data

Demo E1: Adding Data to Tables

Adding a new alignment of super hero where some characters can become anti-hereos.

```
-- This query adds a new row to the alignment table so that some super-heroes
-- can be classed as anti-heroes
INSERT INTO alignment(id, alignment)
VALUES('5', 'Anti Hero');
```

Results | Messages

	id	alignment
1	1	Good
2	2	Bad
3	3	Neutral
4	4	N/A
5	5	Anti Hero

```
-- This query adds a new super hero called killmonger to the table
INSERT INTO superhero (id, superhero_name, full_name, gender_id, eye_colour_id,
hair_colour_id, skin_colour_id, race_id, publisher_id, alignment_id, height_cm,
weight_kg)
VALUES(757, 'Killmonger', 'Erik Killmonger', 1, NULL, NULL, NULL, NULL, 13, 5, 180,
75);
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	757	Killmonger	Erik Killmonger	1	NULL	NULL	NULL	NULL	13	5	180	75

Demo E2: Modifying and Removing Data

Modifying and Removing the Killmonger character that was created in the previous demonstration

```
-- This query edits the fields eye_colour_id and race_id for killmonger
-- in the superhero table
UPDATE superhero
SET eye_colour_id = 3, race_id = 24
WHERE superhero_name = 'Killmonger';
```

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	757	Killmonger	Erik Killmonger	1	NULL	NULL	NULL	NULL	13	5	180	75

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	757	Killmonger	Erik Killmonger	1	3	NULL	NULL	24	13	5	180	75

```
-- This query removes the row from the superhero table that contains the
superhero_name
-- killmonger
DELETE
FROM superhero
WHERE superhero_name = 'Killmonger';
```

100 %

Results | Messages

id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg

Demo E3: Generating Automatic Column Values

Creating a new table with auto-incrementing ID values for each inserted row

```
-- This query creates a table called movie_releases that has an
-- ID field that increments automatically with each insertion
-- of new row data to the table
CREATE TABLE movie_releases (
       movie_id INT IDENTITY(1,1) PRIMARY KEY,
       movie_name VARCHAR(100),
       movie_publisher INT,
       movie_release datetime
);
```

100 %

Results | Messages

movie_id	movie_name	movie_publisher	movie_release

f) Module 8: Using Built-In Functions

Why do programmers use built in functions?

TSQL and programming languages use functions. The biggest reasons are functions allow you to do calculation and break programming into more manageable pieces.

Demo F1: Writing Queries with Built-In Functions

Using built in queries to manipulate data stored in the table

```sql
-- Using the built in functions this statement returns the select
-- statement as upper case
-- lower case
-- Reversed string
-- Unicode Value
SELECT UPPER(full_name) AS [Full Name Uppercase],
LOWER(full_name) AS [Full Name Lowercase],
REVERSE(full_name) AS [Reversed Full Name],
UNICODE(full_name) AS [Unicode]
FROM superhero
```

Results | Messages

	Full Name Uppercase	Full Name Lowercase	Reversed Full Name	Unicode
1	CHARLES CHANDLER	charles chandler	reldnahC selrahC	67
2	RICHARD MILHOUSE JONES	richard milhouse jones	senoJ esuohliM drahciR	82
3	ABRAHAM SAPIEN	abraham sapien	neipaS maharbA	65
4	-	-	-	45
5	EMIL BLONSKY	emil blonsky	yksnolB limE	69
6	ABRAXAS	abraxas	saxarbA	65
7	NULL	NULL	NULL	NULL
8	-	-	-	45
9	ADAM STRANGE	adam strange	egnartS madA	65
10	SHARON CARTER	sharon carter	retraC norahS	83
11	BOB	bob	boB	66
12	CHRISTOPH NORD	christoph nord	droN hpotsirhC	67
13	GABRIEL LAN	gabriel lan	naL leirbaG	71
14	NULL	NULL	NULL	NULL
15	NULL	NULL	NULL	NULL
16	ALEXANDER J. MERCER	alexander j. mercer	recreM .J rednaxelA	65
17	ALEX WOOLSLY	alex woolsly	ylslooW xelA	65
18	ALFRED THADDEUS CRANE PENNYWORTH	alfred thaddeus crane pennyworth	htrowynneP enarC sueddahT derflA	65
19	XENOMORPH	xenomorph	hpromoneX	88

Demo F2: Using Conversion Functions

Converting a string date to a datetime datatype

```sql
-- This query will convert any inputted string of a date
-- into the datetime datatype
SELECT CONVERT(datetime, '2025-02-01')
```

Results | Messages

	(No column name)
1	2025-02-01 00:00:00.000

```sql
-- The TRY_CONVERT function is a fail safe function
-- that will output a NULL value if the conversion is
-- not possible
SELECT TRY_CONVERT(datetime, 'This is a test')
```

Results | Messages

	(No column name)
1	NULL

Demo F3: Using Logical Functions

Using logical functions to perform logical operations on data and output them to the table

```sql
-- This query selects all super hereos that are not aligned to the
-- hero faction
SELECT *
FROM superhero
WHERE alignment_id != 1;

-- This query will tell the database engineer whether or not
-- a value in the table is numerical or not using the
-- ISNUMERIC logical function
SELECT ISNUMERIC(superhero_name)
FROM superhero
WHERE id = 5;

-- This query will filter through all the super heroes in the
-- superhero table and will return a column that checks if the
-- hero is heavy or light weight
SELECT superhero_name, weight_kg, IIF(weight_kg > 150, 'Heavy', 'Light')
       AS weight_class
FROM superhero;
```

100 %

Results | Messages

	id	superhero_name	full_name	gender_id	eye_colour_id	hair_colour_id	skin_colour_id	race_id	publisher_id	alignment_id	height_cm	weight_kg
1	5	Abomination	Emil Blonsky	1	14	1	1	28	13	2	203	441
2	6	Abraxas	Abraxas	1	7	4	1	12	13	2	0	0
3	7	Absorbing Man	NULL	1	7	1	1	24	13	2	193	122
4	13	Air-Walker	Gabriel Lan	1	7	31	1	1	13	2	188	108
5	14	Ajax	NULL	1	9	4	1	13	13	2	193	90
6	16	Alex Mercer	Alexander J. Mercer	1	1	1	1	24	25	2	0	0
7	19	Alien	Xenomorph	1	1	1	4	57	3	2	244	169
8	21	Amazo	-	1	23	1	1	5	4	2	257	173

	(No column name)
1	0

	superhero_name	weight_kg	weight_class
1	3-D Man	90	Light
2	A-Bomb	441	Heavy
3	Abe Sapien	65	Light
4	Abin Sur	90	Light
5	Abomination	441	Heavy
6	Abraxas	0	Light
7	Absorbing Man	122	Light

Query executed successfully.

Demo F4: Using Functions to Work with NULL

Using built in functions to handle NULL values when selecting data

```sql
-- This query uses the function ISNULL to filter data and
-- replace any NULL full names with 'No Full Name'
SELECT superhero_name, ISNULL(full_name, 'No Full Name')
FROM superhero

-- This query will return NULL for height_cm and weight_kg
-- if the values in those columns are 0
SELECT superhero_name, full_name, NULLIF(height_cm, 0) AS height_cm_new,
NULLIF(weight_kg, 0) AS weight_kg_new
FROM superhero;
```

100 %

Results | Messages

	superhero_name	(No column name)
1	3-D Man	Charles Chandler
2	A-Bomb	Richard Milhouse Jones
3	Abe Sapien	Abraham Sapien
4	Abin Sur	-
5	Abomination	Emil Blonsky
6	Abraxas	Abraxas
7	Absorbing Man	No Full Name
8	Adam Monroe	-
9	Adam Strange	Adam Strange

	superhero_name	full_name	height_cm_new	weight_kg_new
1	3-D Man	Charles Chandler	188	90
2	A-Bomb	Richard Milhouse Jones	203	441
3	Abe Sapien	Abraham Sapien	191	65
4	Abin Sur	-	185	90
5	Abomination	Emil Blonsky	203	441
6	Abraxas	Abraxas	NULL	NULL
7	Absorbing Man	NULL	193	122
8	Adam Monroe	-	NULL	NULL
9	Adam Strange	Adam Strange	185	88

Query executed successfully.

Demo G1: Using GROUP BY and Applying Aggregation

A query to count the amount of superhero that are assigned to each gender

```
-- This query counts how many super heroes are assigned to
-- each gender
SELECT COUNT(id) AS [Hero Count], gender_id
FROM superhero
GROUP BY gender_id;
```

	Hero Count	gender_id
1	28	3
2	520	1
3	203	2

Demo G2: Using HAVING and Applying Aggregation

Using the HAVING clause to filter select results

```
-- This query selects the eye colour where the
-- amount of heroes with said I colour is greater
-- than 5
SELECT COUNT(id) AS [Hero Count], eye_colour_id
FROM superhero
GROUP BY eye_colour_id
HAVING COUNT(id) > 5;
```