

DatabaseAndApplicationDev

by Ashleigh Simons

Submission date: 08-Jan-2023 09:35PM (UTC+0000)

Submission ID: 194003489

File name: 23605_Ashleigh_Simons_DatabaseAndApplicationDev_464571_284167037.docx (1.56M)

Word count: 1194

Character count: 6467

Module: Databases and Application Development

Module Code: ACCA5026

Assignment Title: Database development Case study

Student Name: Ashleigh Simons

Student ID Number: 2003948

Contents

Page		Description
3	-	Introduction
4	-	Task 1: Database Creation
		A) i) Tables designed to the 3NF
5	-	A) ii) Entity Relationship Design
6	-	A) iii) Primary Keys
7	-	A) iv) Foreign Keys
8	-	A) v) Check Constraints
9	-	A) vi) Data Performance Tuning Enhancements
10	-	B) i) Insert Statements
11	-	B) ii) Functionality Testing
12	-	C) i) Transaction Testing
13	-	Task 2: Report
		1. Performance Tuning
		1.A) Secondary Indexes
		1.B) Additional Secondary Indexes Added
14	-	2. Distributed Database Solution
		2.A.i) Horizontal fragmentation
		2.B) Reconstruction of global relations
		Reference list

Health and Wealth Hospital

Within task one it will include firstly include designs to the 3NF to show the contents and process of the use of the database. Then it will show an ER diagram representing the databases design to show the links between tables and how information will flow through the database. Furthermore, primary keys will be given and showcased for each of the tables. Additionally, any foreign keys within tables will also be highlighted.

The completion of two check constraints will be explained so that it is understood what they are checking for within the database. Furthermore, within task 2, performance tuning enhancements will be documented. With the intent to explain and justify reasoning of choosing specific tuning enhancements. Finally, there will be a distributed database solution explaining the chosen fragmentation schema strategy and the form of reconstruction of global relations from the fragments.

Task 1: Database Creation

A) i) Tables designed to the 3NF

PatientID	PatientFirstName	PatientSurname	Patient Email	PatientContactNumber	PatientPostcode	PatientAdmissionOutcome
1	Ash	Simons	AS@gmail.com	08757498874	SA18GH	Admitted
2	Sophie	Jones	SJ@hotmail.com	02456343334	SA10GRT	Admitted
3	Tom	Platt	TP@btinternet.com	01283745677	SA189T	Not Admitted

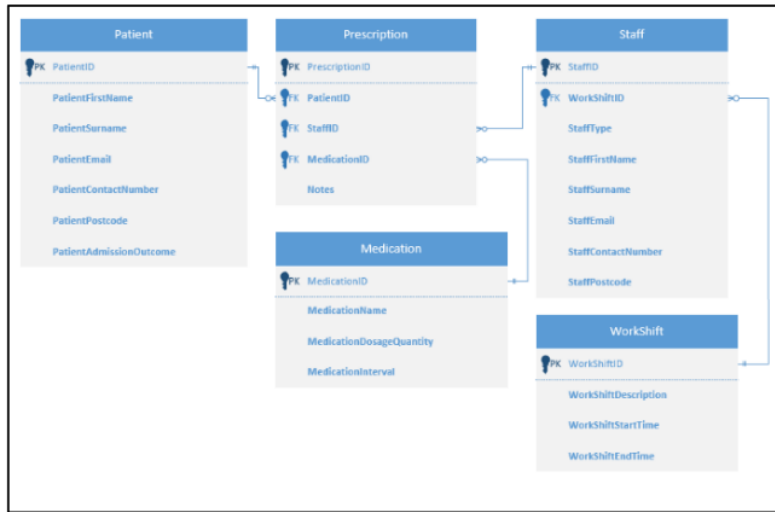
WorkShiftID	WorkShift Description	WorkShiftStartTime	WorkShiftEndTime
1	Morning	08:00	20:00
2	Evening	20:00	08:00

MedicationID	MedicationName	MedicationDosageQuantity	MedicationInterval
1	Paracetamol	250-500mg	4 hours
2	Ibuprofen	200-300mg	4 hours
3	Codeine	15-60mg	4 hours
4	Tramadol	50-150mg	12 hours
5	Morphine	10-20mg	4 hours
6	Aspirin	250-650mg	4 hours
7	Oxycodone	5-15mg	5 hours
8	Gabapentin	450-1800mg	12 hours
9	Fentanyl	12mg	1 hour
10	Ketamine	6-13mg	25 minutes

StaffID	WorkShiftID	StaffType	StaffFirstName	StaffSurname	StaffEmail	StaffContactNumber	StaffPostcode
1	1	Receptionist	Jane	Evans	JE@HWH.com	08763452774	SA10 6HK
2	1	Nurse	Lucy	Mort	LM@HWH.com	02456343267	SA11 JHG
3	1	Nurse	Bill	Peters	BP@HWH.com	08757493844	SA1 7GH
4	2	Doctor	Mark	Davies	MD@HWH.com	01283748588	SA1 9GP
5	2	Doctor	Howard	Hayward	HH@HWH.com	09876657433	SA9 6KL

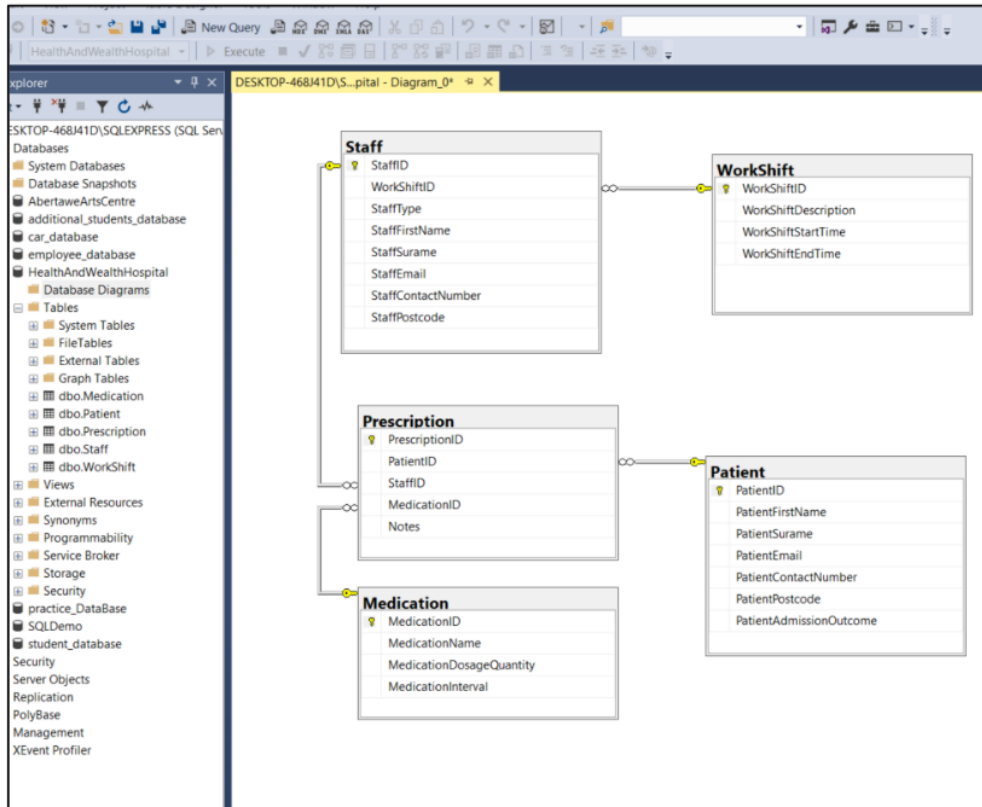
PrescriptionID	PatientID	StaffID	MedicationID	Notes
1	1	5	4	First dose at 13:45
2	2	5	7	First dose at 06:00
3	3	4	3	First dose at 08:50
4	1	4	4	First dose at 19:00
5	1	4	1	First dose at 23:00
6	3	5	1	First dose at 01:00
7	3	5	6	First dose at 09:30

A) ii) Entity Relationship Design



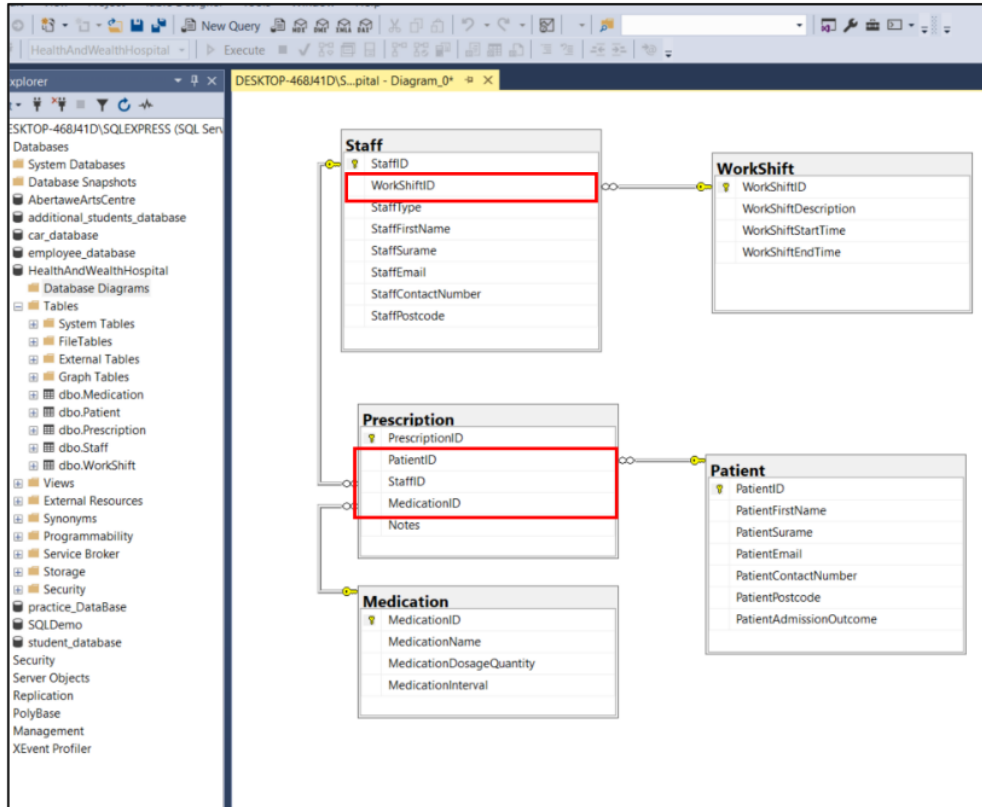
A) iii) Primary Keys

Visual example of what is included within the files.



A) iv) Foreign Keys

Visual example of what is included within the files.



A) v) Check Constraints

Examples of what is included within the files.

Patient Table

```
-- Now the creations of the tables.
-- Create the Patient Table and Attributes.
CREATE TABLE Patient(
    PatientID          VARCHAR(6) NOT NULL CONSTRAINT PK_Patient PRIMARY KEY,
    PatientFirstName   VARCHAR(30) NOT NULL,
    PatientSurname     VARCHAR(30) NOT NULL,
    PatientEmail       VARCHAR(50) NOT NULL,
    PatientContactNumber VARCHAR(11) NOT NULL CONSTRAINT CHK_Patient_PatientContactNumber CHECK (PatientContactNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    PatientPostcode    VARCHAR(8) NOT NULL,
    PatientAdmissionOutcome VARCHAR(15) NOT NULL,
);
GO
```

```
PatientContactNumber VARCHAR(11) NOT NULL CONSTRAINT CHK_Patient_PatientContactNumber
```

```
CHECK (PatientContactNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
```

Staff Table

```
StaffContactNumber VARCHAR(11) NOT NULL CONSTRAINT CHK_Staff_StaffContactNumber CHECK (StaffContactNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
```

```
StaffContactNumber VARCHAR(11) NOT NULL CONSTRAINT CHK_Staff_StaffContactNumber
```

```
CHECK (StaffContactNumber LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
```

A) vi) Data Performance Tuning Enhancements

Examples of what is included within the files.

```
-- PERFORMANCE TUNING  
CREATE INDEX idx_medicationDosageQuantity ON Medication (MedicationDosageQuantity)  
CREATE INDEX idx_medicationID ON Prescription (MedicationID)
```

B) i) Insert Statements

One Example of many included in the files.

```
-- Enter data into each table that we require.
-- Data for the Patient Table.
INSERT INTO Patient([PatientID], [PatientFirstName], [PatientSurname], [PatientEmail], [PatientContactNumber], [PatientPostcode], [PatientAdmissionOutcome])
-- VALUES are the specified data that is used within the fields highlighted.
VALUES ('1', 'Ash', 'Simons', 'ash.simons@gmail.com', '08763452774', 'SA18GH', 'Admitted'),
('2', 'Sophie', 'Jones', 'sophie.jones@hotmail.com', '02456343334', 'SA10GRT', 'Admitted'),
('3', 'John', 'Smith', 'john.smith@gmail.com', '12345678901', 'SW1A2AA', 'Admitted'),
('4', 'Jane', 'Doe', 'jane.doe@yahoo.com', '12345678901', 'N1C4DA', 'Admitted'),
('5', 'Michael', 'Johnson', 'michael.johnson@outlook.com', '12345678901', 'EC2A3ET', 'Admitted'),
('6', 'Samantha', 'Williams', 'samantha.williams@hotmail.com', '12345678901', 'W1B3CY', 'Admitted'),
('7', 'Jacob', 'Brown', 'jacob.brown@gmail.com', '12345678901', 'E1A2B3', 'Admitted'),
('8', 'Alex', 'Jones', 'alex.jones@gmail.com', '12345678901', 'SE1A1AA', 'Admitted'),
('9', 'Emily', 'Davis', 'emily.davis@yahoo.com', '12345678901', 'M1C5DE', 'Admitted'),
('10', 'Ryan', 'Miller', 'ryan.miller@outlook.com', '12345678901', 'W1F7GY', 'Not Admitted'),
('11', 'Emma', 'Garcia', 'emma.garcia@hotmail.com', '12345678901', 'E2D4F5', 'Admitted'),
('12', 'Ben', 'Rodriguez', 'ben.rodriguez@gmail.com', '12345678901', 'N3C6G7', 'Admitted'),
('13', 'Olivia', 'Martinez', 'olivia.martinez@gmail.com', '12345678901', 'NW1A2AB', 'Not Admitted'),
('14', 'Ava', 'Jackson', 'ava.jackson@yahoo.com', '12345678901', 'EC1A3EF', 'Not Admitted'),
('15', 'Isabella', 'Thompson', 'isabella.thompson@outlook.com', '12345678901', 'WC2B4CG', 'Admitted')
GO
```

	PatientID	PatientFirstName	PatientSurname	PatientEmail	PatientContactNumber	PatientPostcode	PatientAdmissionOutcome
1	1	Ash	Simons	ash.simons@gmail.com	08763452774	SA18GH	Admitted
2	10	Ryan	Miller	ryan.miller@outlook.com	12345678901	W1F7GY	Not Admitted
3	11	Emma	Garcia	emma.garcia@hotmail.com	12345678901	E2D4F5	Admitted
4	12	Ben	Rodriguez	ben.rodriguez@gmail.com	12345678901	N3C6G7	Admitted
5	13	Olivia	Martinez	olivia.martinez@gmail.com	12345678901	NW1A2AB	Not Admitted
6	14	Ava	Jackson	ava.jackson@yahoo.com	12345678901	EC1A3EF	Not Admitted
7	15	Isabella	Thompson	isabella.thompson@outlook.com	12345678901	WC2B4CG	Admitted
8	2	Sophie	Jones	sophie.jones@hotmail.com	02456343334	SA10GRT	Admitted
9	3	John	Smith	john.smith@gmail.com	12345678901	SW1A2AA	Admitted
10	4	Jane	Doe	jane.doe@yahoo.com	12345678901	N1C4DA	Admitted
11	5	Michael	Johnson	michael.johnson@outlook.com	12345678901	EC2A3ET	Admitted
12	6	Samantha	Williams	samantha.williams@hotmail.com	12345678901	W1B3CY	Admitted
13	7	Jacob	Brown	jacob.brown@gmail.com	12345678901	E1A2B3	Admitted
14	8	Alex	Jones	alex.jones@gmail.com	12345678901	SE1A1AA	Admitted
15	9	Emily	Davis	emily.davis@yahoo.com	12345678901	M1C5DE	Admitted

B) ii) Functionality Testing

```
SELECT * FROM Patient WHERE PatientID LIKE '4';
```

89 %

Results Messages

	PatientID	PatientFirstName	PatientSurname	PatientEmail	PatientContactNumber	PatientPostcode	PatientAdmissionOutcome
1	4	Jane	Doe	jane.doe@yahoo.com	12345678901	N1C4DA	Admitted

```
SELECT * FROM WorkShift WHERE WorkShiftDescription LIKE 'Evening';
```

89 %

Results Messages

	WorkShiftID	WorkShiftDescription	WorkShiftStartTime	WorkShiftEndTime
1	2	Evening	20:00	08:00

```
SELECT * FROM Medication WHERE MedicationInterval LIKE '4 hours';
```

89 %

Results Messages

	MedicationID	MedicationName	MedicationDosageQuantity	MedicationInterval
1	1	Paracetamol	250-500mg	4 hours
2	2	Ibuprofen	200-300mg	4 hours
3	3	Codeine	15-60mg	4 hours
4	5	Morphine	10-20mg	4 hours
5	6	Asprin	250-650mg	4 hours

```
SELECT * FROM Staff WHERE StaffType LIKE 'Doctor';
```

89 %

Results Messages

	StaffID	WorkShiftID	StaffType	StaffFirstName	StaffSurname	StaffEmail	StaffContactNumber	StaffPostcode
1	10	1	Doctor	Ben	Rodriguez	ben.rodriguez@gmail.com	12345678901	N3C6G7
2	13	2	Doctor	Ava	Jackson	ava.jackson@yahoo.com	12345678901	EC1A3EF
3	14	1	Doctor	Isabella	Thompson	isabella.thompson@outlook.com	12345678901	WC2B4CG
4	15	2	Doctor	Sophia	Lopez	sophia.lopez@hotmail.com	12345678901	E3D5F6
5	16	1	Doctor	Liam	Gonzalez	liam.gonzalez@gmail.com	12345678901	N5C8G9
6	9	2	Doctor	Emma	Garcia	emma.garcia@hotmail.com	12345678901	E2D4F5

```
SELECT * FROM Prescription WHERE PatientID LIKE '2';
```

89 %

Results Messages

	PrescriptionID	PatientID	StaffID	MedicationID	Notes
1	2	2	16	1	First dose at 12:15

C) i) Transaction Testing

```
-- Now to create a prescription for a patient.
-- Prescription 1.
-- We need to start a transaction to create a prescription.
BEGIN TRANSACTION CreateNewPrescription
BEGIN TRY -- used to catch errors that may occur.

-- Create a new prescription with all of the values required for making a prescription.
INSERT INTO [Prescription] (PrescriptionID, PatientID, StaffID, MedicationID, Notes)
VALUES ('1','1','10','5','First dose at 10:45')

-- Now this can be confirmed by committing the transaction.
COMMIT TRANSACTION CreateNewPrescription;
END TRY

-- Adding a catch to make sure if there are errors it will be highlighted.
BEGIN CATCH
-- An error has occurred.
-- In addition to the CATCH a THROWBACK of the transaction is used to see if
-- there are any additional errors with the transaction.
ROLLBACK TRANSACTION CreateNewPrescription;
PRINT('Rolling back, there was an error.');
```

THROW;

END CATCH;

```
-- Testing SQL Code for the rollback transaction (Invalid Prescription ID)
BEGIN TRANSACTION CreateNewPrescription
BEGIN TRY -- used to catch errors that may occur.
INSERT INTO [Prescription] (PrescriptionID, PatientID, StaffID, MedicationID, Notes)
VALUES ('12345678910','4','9','5','First dose at 19:00')
COMMIT TRANSACTION CreateNewPrescription;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION CreateNewPrescription;
PRINT('Rolling back, there was an error.');
```

THROW;

END CATCH;

0 %

Messages

(0 rows affected)
Rolling back, there was an error.
Msg 2628, Level 16, State 1, Line 144
String or binary data would be truncated in table 'HealthShed\HealthHospital.dbo.Prescription', column 'PrescriptionID'. Truncated value: '12345678910'.

Completion time: 2023-01-09T13:55:24.212367+00:00

Task 2: Report

1. Performance Tuning

1.A) Secondary Indexes

Within Task 1.A.vi) there were two secondary indexes created and added to the database to improve the performance of the data queries that would be completed within the database. A secondary index is a form of improving the databases query speeds. Their purpose can be described as a means to “quicken the search by reducing the number of records to search for” [1]. This implies that making a generalized search within a table or multiple tables for information whilst looking at specific records which also contain something within the index given could reduce the time required to search for acceptable results.

Specifically, the reasoning for implementing the first of the two indexes was to target queries involving patient medication, dosage quantity being at focus. After studying the brief it was highlighted that the nurses would be caring for several patients and providing them with their medication. Furthermore, with the amount of responsibility and importance of this process and expected usage of the database for this role. Supporting for the need of this secondary index to be created. Since having this secondary index implemented it will hope to cut down on time that would be spent looking for information on the database, and help the hospital provide care to patients more efficiently, whilst also maintaining its accuracy.

Next, the second of the two indexes added is for the medication ID within the prescription. Debatably another very important index to be added, as it could work within the process of the queries involving the previous index created. When the prescriptions are created, and it comes to the nurses providing the care to the patients, they will be able to search for how many of the patients are taking the same medication. This could not only be used to track inventory but ensure that there is enough of the medication for the number of patients prescribed within the hospital. It could also serve as a means of being more efficient and organized within the hospital and departments. If the staff are aware that for instance 10 patients are taking the same medication, they would be able to prepare that medication in advance, so that when the patients are allowed to receive their doses it is all prepared for them.

If used correctly and in advance for some purposes, these indexes could provide a quicker and more effective way of querying through the database. Furthermore, speeding up the time it takes for the nurses to provide care to the patients.

1.B) Additional Secondary Indexes Added

Please look at Task 1.A.vi) to see an image of the additional secondary indexes added to the SQL file 1.

2. Distributed Database Solution

2.A.i) Horizontal fragmentation

Horizontal fragmentation schema has been decided as the proposed schema of this database as the tables have been designed to be smaller with specific fields for each table. The benefits to using this schema include “dividing up the workload across multiple smaller servers” [2]. This suggests that each server will be accountable for each of the tables and this has an advantage as you can add more hardware and spend more money on resources for the aspects that require it more than others. This allows you to be smarter with spending and resources so that you only spend what you need to when you need to.

A few negatives to this approach is “power outages, failure with regular updates, and physical hardware configuration issues” [3]. These imply that the more decides you have the more individual issues that you could be encountered with. However, if you have one server that encounters these problems then you might not have access to any of the system so everything could be halted rather than some aspects of the whole system.

2.B) Reconstruction of global relations

To be able to reconstruct the global relations you would have to collect all of the tables that have been created. Then the tables would have to be looked at so that patterns and connections can be identified [4]. These can be in the form of themes, names, information within the tables that could possibly allow them to fit together. Once some of these have been found and decided the tables and the contents can be organized and then grouped together. Now reconstruction can be completed, and verification of information should be done so that small errors are not encountered. If they are then it could cause bigger problems and could result in information, details, and general data being lost or damaged.

Reference list

- [1] K. Nath, “An in-depth look at database indexing,” *freeCodeCamp.org*, 28-Apr-2021. [Online]. Available: <https://www.freecodecamp.org/news/database-indexing-at-a-glance-bb50809d48bd/>
- [2] J. Kehayias, “Pros and cons of horizontal vs. vertical scaling,” *SolarWinds - Database Performance Monitoring & DataOps*. [Online]. Available: <https://www.sentryone.com/blog/pros-and-cons-of-horizontal-vs-vertical-scaling>
- [3] J. Borgini, “5 common server issues and their effects on operations: TechTarget,” *Data Center*, 10-Jun-2020. [Online]. Available: <https://www.techtarget.com/searchdatacenter/tip/5-common-server-issues-and-their-effects-on-operations>
- [4] SQLTreeo, “Fragmentation and defragmentation in SQL server,” *SQLTreeo*, 29-Apr-2021. [Online]. Available: <https://www.sqltreeo.com/docs/fragmentation-and-defragmentation-in-sql-server>

FINAL GRADE

94/100

GENERAL COMMENTS

Instructor

Hi Ashleigh,

You got 94%
. Excellent work.

Normalization
was well attempted clearly showing the steps. This
was a very critical task for
the next sections. The evidence of the practical
work carried out was clearly
presented.

Referencing
and Citations are vital in report writing and you
have adhered to IEEE format
of referencing.

I hope
you will continue giving your best effort in the
forthcoming assessments.

Should
you have any questions let me know

All the
Best

Regards

Divya

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14
