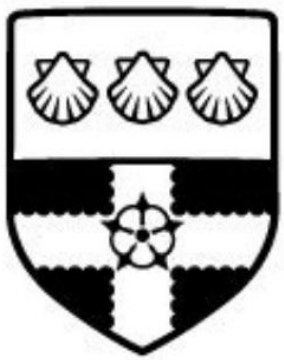


Group **XIV**

“Destinations Database” Group **Report**

CS2DI19: Databases

University of Reading

Created by:

Jason Dookarun: 26017434

Shavin Croos: 27015244

Abdullah Rana: 28005792

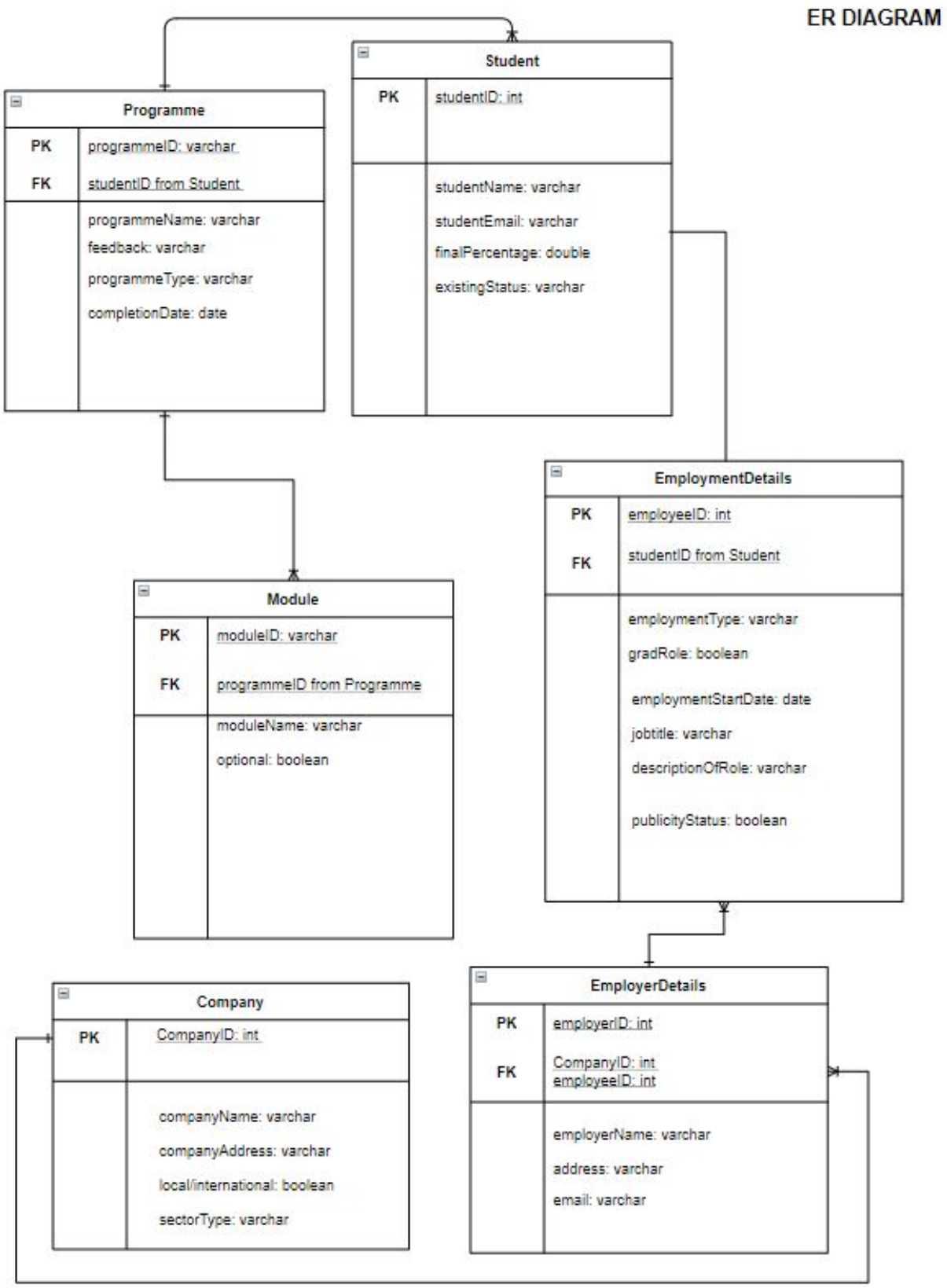
Date of Completion: 18th November 2020

Actual hrs spent: 25 Hour(s)

Assignment Evaluation: A very creative project which encouraged problem solving and critical thinking to ensure our database is modelled in an appropriate manner. A positive aspect was the team-working it encouraged. An improvement could perhaps be to allow a wider range of scenarios to model a database upon.

Table of Contents

I: ER Diagram


ER DIAGRAM
Programme
PK
programmeID: varchar
FK
studentID from Student
programmeName: varchar
feedback: varchar
programmeType: varchar
completionDate: date
Student
PK
studentID: int
studentName: varchar
studentEmail: varchar
finalPercentage: double
existingStatus: varchar
Module
PK
moduleID: varchar
FK
programmeID from Programme
moduleName: varchar
optional: boolean
EmploymentDetails
PK
employeeID: int
FK
studentID from Student
employmentType: varchar
gradRole: boolean
employmentStartDate: date
jobtitle: varchar
descriptionOfRole: varchar
publicityStatus: boolean
Company
PK
CompanyID: int
companyName: varchar
companyAddress: varchar
local/international: boolean
sectorType: varchar
EmployerDetails
PK
employerID: int
FK
CompanyID: int
employeeID: int
employerName: varchar
address: varchar
email: varchar

II: Normalisation

For a database to be in Third Normal Form (3NF) it is required to be in both Second Normal Form (2NF) and First Normal Form (1NF) alongside not having Partial and Transitive dependencies. Thus, our database can be justified to be 3NF as follows:

1st Normal Form

Firstly, all our tables only have atomic valued attributes/columns. For example, in the student table, for one ID it will only have one student name, email and final percentage. It is structured in a way such that no data entered is repeated in the table. Furthermore, all values stored in each column are of the same domain - student IDs are all type “int”, names, emails are stored as “varchar” and final marks are of type “double”. Moreover, the columns of each table have names which are unique; none are repeated, and there is no requirement for the data to be entered in a particular order, preventing any confusion or ambiguity in the system. As a result of meeting these criteria, our database can be said to be in 1st Normal Form.

2nd Normal Form

Now, for our tables to be in 2NF, it must not have any form of partial dependency. Partial dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key **[1]**. The problem of partial dependency can be solved if a table is divided, an attribute that causes partial dependency is removed, and is moved to another table where it would fit best. In our initial 1NF table, it contained attributes such as CompanyEmail in the student table. However, this detail was only dependent on companyID and not studentID since the email is that of the company. So to counteract this, we created a new employee table for employee specific details, and moved the CompanyEmail to there. By analysing all tables for partial dependencies, and making necessary changes, we can confidently say our table is in 2NF; no attributes in any table only rely on part of a primary key.

3rd Normal Form

Lastly, to achieve 3NF, our database now must not contain transitive dependency. Transitive dependency is when a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key **[2].** An example would be in a previous iteration of our database model, where we had the attribute IncludedModules stored under the programme table. The issue was this attribute wasn’t solely dependent on the primary key; it was also dependent on non-prime attributes such as ProgrammeType and ProgrammeName. To rectify this issue, we actually removed the IncludedModules attribute. Instead we created a module table and defined a different form of relationship between Programme and Module - where one programme may have one or many modules (as shown in the ER diagram above). We applied these principles to all tables to ensure that no table contains transitive dependency, thus proving our database to be in 3rd Normal Form.

III: Table Creation & Data Injection

Table Name: Module

Table Creation	Data Injection
CREATE TABLE MODULES1(MODULEID VARCHAR(255), MODULENAME VARCHAR(255), OPTIONAL BOOLEAN, PROGRAMMEID VARCHAR(255), **CONSTRAINT** FK_PROGRAMMEID FOREIGN KEY (PROGRAMMEID) REFERENCES PROGRAMME1 (PROGRAMMEID), **CONSTRAINT** PK_MODULES1 PRIMARY KEY (MODULEID));	***insert into*** *MODULES1 (MODULEID, MODULENAME, OPTIONAL, PROGRAMMEID) values ('CS2JA16', 'PROGRAMMING IN JAVA', TRUE, 'CS2020NX');* ***insert into*** *MODULES1 (MODULEID, MODULENAME, OPTIONAL, PROGRAMMEID) values ('FS2NT17', 'NUTRITION', FALSE, 'FS2020TY');* ***insert into*** *MODULES1 (MODULEID, MODULENAME, OPTIONAL, PROGRAMMEID) values ('BM2CB17', 'CONSUMER BEHAVIOUR', TRUE, 'BM2020GL');*

Evidence

Query Editor Query History

```
1  select * from modules1
2
```

Data Output Explain Messages Notifications

	moduleid [PK] character varying (255)	modulename character varying (255)	optional boolean	programmeid character varying (255)
1	CS2JA16	PROGRAMMING IN JAVA	true	CS2020NX
2	FS2NT17	NUTRITION	false	FS2020TY
3	BM2CB17	CONSUMER BEHAVIOUR	true	BM2020GL
4	EC2ME16	MICROECONOMICS	true	EC2020HV
5	GD2DT18	DESIGN THINKING	false	GD2020IG
6	MK2BT17	BUSINESS THINKING	false	MK2020WD
7	MT2MM16	MECHANICAL MATHS	true	MT2020SX
8	CM2PM17	PHYSICAL PROCESSES & MO...	true	CM2020OJ
9	BS2BC16	BIOCHEMISTRY	false	BS2020AY
10	EG2AP17	AERONAUTICAL PHYSICS	true	EG2020QL

Table Name: Student

Table Creation	Data Injection
CREATE TABLE STUDENT1(STUDENTID INTEGER, STUDENTNAME VARCHAR(255), STUDENTEMAIL VARCHAR(255), FINALPERCENTAGE DOUBLE, EXISTINGSTATUS VARCHAR(255), **CONSTRAINT** PK_STUDENT1 PRIMARY KEY (STUDENTID));	***insert into*** *STUDENT1 (STUDENTID, STUDENTNAME, STUDENTEMAIL, FINALPERCENTAGE, EXISTINGSTATUS) values (29409552, 'SAMUEL JONES', 's.jones@student.reading.ac.uk', 90.4, 'Undergraduate');* ***insert into*** *STUDENT1 (STUDENTID, STUDENTNAME, STUDENTEMAIL, FINALPERCENTAGE, EXISTINGSTATUS) values (26347199, 'ANNA FELICITY', 'a.felicity@student.reading.ac.uk', 85.7, 'Undergraduate');* ***insert into*** *STUDENT1 (STUDENTID, STUDENTNAME, STUDENTEMAIL, FINALPERCENTAGE, EXISTINGSTATUS) values (27673198, 'TOM JACKSON', 't.jackson@student.reading.ac.uk', 88.3, 'Undergraduate');*

Evidence

```
1  select * from student1
```

Data Output | Explain | Messages | Notifications

	studentid [PK] integer	studentname character varying (255)	studentemail character varying (255)	finalpercentage double precision	existingstatus character varying (255)
1	29409552	SAMUEL JONES	s.jones@student.reading.ac.uk	90.4	Undergraduate
2	26347199	ANNA FELICITY	a.felicity@student.reading.ac....	85.7	Undergraduate
3	27673198	TOM JACKSON	t.jackson@student.reading.ac...	88.3	Undergraduate
4	12891283	SAMUEL L JACKSON	s.l.jackson@student.reading.a...	99.8	Undergraduate
5	43739173	JAMES BOND	j.bond@student.reading.ac.uk	90.2	Undergraduate
6	28281711	JEFF SHREEVES	j.shreeves@student.reading.a...	81.2	Undergraduate
7	11182212	ALAN SMITH	alan.smith@student.reading.a...	78.2	Undergraduate
8	18187221	CHRIS KAMARA	c.kamara@student.reading.ac...	89	Postgraduate
9	98287112	MARTIN TYLER	j.shreeves@student.reading.a...	81.2	Undergraduate
10	69696969	EMILE HESKEY	e.heskey@student.reading.ac....	98.2	Undergraduate

Table Name: Programme

Table Creation

```
CREATE TABLE PROGRAMME1(
        PROGRAMMEID VARCHAR(255),
        PROGRAMMENAME VARCHAR(255),
        FEEDBACK VARCHAR(255),
        PROGRAMMETYPE VARCHAR(255),
        COMPLETIONDATE DATE,
        STUDENTID INTEGER,

CONSTRAINT FK_STUDENTID FOREIGN KEY
(STUDENTID) REFERENCES STUDENT1
(STUDENTID),

CONSTRAINT PK_PROGRAMME1 PRIMARY KEY
(PROGRAMMEID));
```

Data Injection

```
insert into PROGRAMME1 (PROGRAMMEID,
PROGRAMMENAME, FEEDBACK,
PROGRAMMETYPE, COMPLETIONDATE,
STUDENTID) values ('CS2020NX', 'COMPUTER
SCIENCE', 'GOOD, BUT THE SUPPORT FOR THIS
PROGRAMME COULD BE BETTER.', 'WITH
INDUSTRIAL YEAR', DATE '2023-05-20', 29409552);

insert into PROGRAMME1 (PROGRAMMEID,
PROGRAMMENAME, FEEDBACK,
PROGRAMMETYPE, COMPLETIONDATE,
STUDENTID) values ('FS2020TY', 'FOOD SCIENCE',
'SOME OF THE MODULES WERE OK, BUT THE
OTHERS WERE NOT AS GOOD.', 'WITHOUT
INDUSTRIAL YEAR', DATE '2023-05-17', 26347199);

insert into PROGRAMME1 (PROGRAMMEID,
PROGRAMMENAME, FEEDBACK,
PROGRAMMETYPE, COMPLETIONDATE,
STUDENTID) values ('BM2020GL', 'BUSINESS
MANAGEMENT', 'FOUND IT TOO EASY, THE
SUPPORT WAS PRETTY GOOD.', 'WITH
INDUSTRIAL YEAR',  DATE '2023-05-19', 27673198);
```

Evidence

Query Editor | Query History

```
1  select * from programme1
```

Data Output | Explain | Messages | Notifications

	programmeid [PK] character varying (255)	programmename character varying (255)	feedback character varying (255)	programmetype character varying (255)	completiondate date	studentid integer
1	CS2020NX	COMPUTER SCIENCE	GOOD, BUT THE SUPPORT FOR THIS PROGR...	WITH INDUSTRIAL YEAR	2023-05-20	29409552
2	FS2020TY	FOOD SCIENCE	SOME OF THE MODULES WERE OK, BUT THE...	WITHOUT INDUSTRIAL YEAR	2023-05-17	26347199
3	BM2020GL	BUSINESS MANAGEMENT	FOUND IT TOO EASY, THE SUPPORT WAS PR...	WITH INDUSTRIAL YEAR	2023-05-19	27673198
4	EC2020HV	ECONOMICS	WAS OK, BUT COULD HAVE BEEN BETTER.	WITH INDUSTRIAL YEAR	2023-05-15	12891283
5	GD2020IG	GRAPHIC COMMUNICATION ...	REALLY ENJOYED IT, WOULD DO IT AGAIN.	WITHOUT INDUSTRIAL YEAR	2023-05-18	43739173
6	MK2020WD	MARKETING	IT WAS INTERESTING HOW THE MARKETS I...	WITHOUT INDUSTRIAL YEAR	2023-05-17	28281711
7	MT2020SX	MATHEMATICS	ALLOWED FOR THE STUDENTS TO COME UP...	WITH INDUSTRIAL YEAR	2023-05-19	11182212
8	CM2020OJ	CHEMISTRY	WASN'T THAT GOOD, THE LECTURER HADN'...	WITH INDUSTRIAL YEAR	2023-05-27	18187221
9	EG2020QL	ENGINEERING	VERY GOOD, REALLY LEARNT ALOT FROM S...	WITH INDUSTRIAL YEAR	2023-05-20	98287112
10	BS2020AY	BIOLOGICAL SCIENCES	IT WAS OK, BUT I FELT THERE COULD HAVE ...	WITHOUT INDUSTRIAL YEAR	2023-05-24	69696969

Table Name: EmploymentDetails

Table Creation

```
CREATE TABLE EMPLOYMENTDETAILS1 (
        EMPLOYEEID INTEGER,
        EMPLOYMENTTYPE VARCHAR(255),
        GRADROLE BOOLEAN,
        EMPLOYMENTSTARTDATE DATE,
        JOBTITLE VARCHAR(255),
        DESCRIPTIONOFROLE VARCHAR(255),
        PUBLICITYSTATUS BOOLEAN,
        STUDENTID INTEGER,

CONSTRAINT FK_STUDENTID FOREIGN KEY
(STUDENTID) REFERENCES STUDENT1
(STUDENTID),

CONSTRAINT PK_EMPLOYEMENTDETAILS1
PRIMARY KEY (EMPLOYEEID));
```

Data Injection

```
insert into EMPLOYMENTDETAILS1
(EMPLOYEEID, EMPLOYMENTTYPE, GRADROLE,
EMPLOYMENTSTARTDATE, JOBTITLE,
DESCRIPTIONOFROLE, PUBLICITYSTATUS,
STUDENTID)
values (241693, 'FULL-TIME', TRUE, DATE
'2024-09-10', 'SOFTWARE DEVELOPER',
'DEVELOPS AND IMPROVES THE SOFTWARE OF
THE COMPANY.', FALSE, 29409552);

insert into EMPLOYMENTDETAILS1
(EMPLOYEEID, EMPLOYMENTTYPE, GRADROLE,
EMPLOYMENTSTARTDATE, JOBTITLE,
DESCRIPTIONOFROLE, PUBLICITYSTATUS,
STUDENTID)
values (185920, 'PART-TIME', FALSE, DATE
'2023-09-17', 'FOOD QUALITY INSPECTOR',
'CHECKS TO SEE IF FOOD MEETS COMPANY
STANDARDS.', FALSE, 26347199);

insert into EMPLOYMENTDETAILS1
(EMPLOYEEID, EMPLOYMENTTYPE, GRADROLE,
EMPLOYMENTSTARTDATE, JOBTITLE,
DESCRIPTIONOFROLE, PUBLICITYSTATUS,
STUDENTID)
values (205318, 'FIXED', TRUE, DATE '2024-09-04',
'WAREHOUSE MANAGER', 'MAKES SURE THE
WAREHOUSE FOR THE GOOD IS ORGANISED.',
TRUE, 27673198);
```

Evidence

Query Editor | Query History

```
1  select * from employmentdetails1
2
```

Data Output | Explain | Messages | Notifications

	employeeid [PK] integer	employmenttype character varying (255)	gradrole boolean	employmentstartdate date	jobtitle character varying (255)	descriptionofrole character varying (255)	publicitystatus boolean	studentid integer
1	241693	FULL-TIME	true	2024-09-10	SOFTWARE DEVELOPER	DEVELOPS AND IMPROVES T...	false	29409552
2	185920	PART-TIME	false	2023-09-17	FOOD QUALITY INSPECTOR	CHECKS TO SEE IF FOOD ME...	false	26347199
3	205318	FIXED	true	2024-09-04	WAREHOUSE MANAGER	MAKES SURE THE WAREHOU...	true	27673198
4	290590	FULL-TIME	true	2023-09-20	ECONOMIST	ANALYSES DATA AND RESEA...	true	12891283
5	667180	PART-TIME	false	2023-08-28	WEB DESIGNER	DEVELOPS AND DESIGNS WE...	false	43739173
6	423690	FIXED	true	2024-08-29	MARKET RESEARCHER	COLLECT AND ANALYSE DAT...	false	28281711
7	330795	FULL-TIME	false	2023-09-05	DATA ANALYST	CHECKS DATA QUALITY AND ...	true	11182212
8	149333	FIXED	false	2023-09-15	ANALYTICAL CHEMIST	ASSESS CHEMICAL STRUCTU...	false	18187221
9	245172	PART-TIME	true	2024-08-27	ELECTRONICS ENGINEER	DESIGNS, PRODUCES, INSTAL...	true	98287112
10	898131	PART-TIME	false	2023-08-29	ENVIRONMENTAL MANAGER	OVERSEES ENVIRONMENTAL ...	true	69696969

Table Name: Company

Table Creation	Data Injection
CREATE TABLE COMPANY1 (COMPANYID INTEGER, COMPANYNAME VARCHAR(255), COMPANYADDRESS VARCHAR(255), LOC BOOLEAN, SECTORTYPE VARCHAR(255), **CONSTRAINT** PK_COMPANY1 PRIMARY KEY (COMPANYID));	***insert into*** *COMPANY1 (COMPANYID, COMPANYNAME, COMPANYADDRESS, LOC, SECTORTYPE) values (467382, 'AMAZON', '87 RAYMOND AVENUE', FALSE, 'E-COMMERCE');* ***insert into*** *COMPANY1 (COMPANYID, COMPANYNAME, COMPANYADDRESS, LOC, SECTORTYPE) values (284910, 'BEBO', '45 GODLY LANE', TRUE, 'FOOD');* ***insert into*** *COMPANY1 (COMPANYID, COMPANYNAME, COMPANYADDRESS, LOC, SECTORTYPE) values (810529, 'SINBAD LTD', '36 HAMMERSMITH RD', TRUE, 'TECHNOLOGY');*

Evidence

Query Editor | Query History

```
1  select * from company1
2
```

Data Output | Explain | Messages | Notifications

	companyid [PK] integer	companyname character varying (255)	companyaddress character varying (255)	loc boolean	sectortype character varying (255)
1	467382	AMAZON	87 RAYMOND AVENUE	false	E-COMMERCE
2	284910	BEBO	45 GODLY LANE	true	FOOD
3	810529	SINBAD LTD	36 HAMMERSMITH RD	true	TECHNOLOGY
4	758968	SANTANDER	56 CHELWOOD AVENUE	false	ECONOMY
5	762874	MICROSOFT	49 BECKWOOD STREET	true	TECHNOLOGY
6	194946	SONY	85 UTHMAN RD	false	TECHNOLOGY
7	957234	ORACLE	84 AKMINSTER DRIVE	true	TECHNOLOGY
8	202885	XLABS	23 NIGHTGOOD LANE	false	SCIENCE
9	532993	APPLE	24 DRANGE CLOSE	false	TECHNOLOGY
10	402069	EVOLITE	9 DOOLEY WAY	true	SCIENCE

Table Name: EmployerDetails

Table Creation

```
CREATE TABLE EMPLOYERDETAILS1 (
        EMPLOYERID INTEGER,
        EMPLOYERNAME VARCHAR(255),
        ADDRESS VARCHAR(255),
        EMAIL VARCHAR(255),
        COMPANYID INTEGER,
        EMPLOYEEID INTEGER,

CONSTRAINT FK_COMPANYID FOREIGN KEY
(COMPANYID) REFERENCES COMPANY1
(COMPANYID),

CONSTRAINT FK_EMPLOYEEID FOREIGN KEY
(EMPLOYEEID) REFERENCES
EMPLOYMENTDETAILS1 (EMPLOYEEID),

CONSTRAINT PK_EMPLOYERDETAILS1
PRIMARY KEY (EMPLOYERID));
```

Data Injection

```
insert into EMPLOYERDETAILS1 (EMPLOYERID,
EMPLOYERNAME, ADDRESS, EMAIL,
COMPANYID, EMPLOYEEID) values (194756, 'JEFF
RAMOS', '8 GREEN LANE', 'j.ramos@mail.com',
467382, 205318);

insert into EMPLOYERDETAILS1 (EMPLOYERID,
EMPLOYERNAME, ADDRESS, EMAIL,
COMPANYID, EMPLOYEEID) values (295184,
'SALLY ATKINS', '25 VICTORIA AVENUE',
's.atkins@yahoo.com', 810529, 241693);

insert into EMPLOYERDETAILS1 (EMPLOYERID,
EMPLOYERNAME, ADDRESS, EMAIL,
COMPANYID, EMPLOYEEID) values (638502,
'JONATHAN CONNORS', '67 HUFFINGTON
STREET', 'j.connors@hotmail.com', 284910,
185920);
```

Evidence

Query Editor | Query History

```
1  select * from employerdetails1
2
```

Data Output | Explain | Messages | Notifications

	employerid [PK] integer	employername character varying (255)	address character varying (255)	email character varying (255)	companyid integer	employeeid integer
1	194756	JEFF RAMOS	8 GREEN LANE	j.ramos@mail.com	467382	205318
2	295184	SALLY ATKINS	25 VICTORIA AVENUE	s.atkins@yahoo.com	810529	241693
3	638502	JONATHAN CONNORS	67 HUFFINGTON STREET	j.connors@hotmail.com	284910	185920
4	236738	JOHNSON BRAVE	3 DOUGHT LANE	j.brave@gmail.com	758968	290590
5	922503	HUW EDWARDS	67 ASLOT RD	h.edwards@mail.com	762874	667180
6	673047	JOEL COURT	34 DORKING AVENUE	j.court@yahoo.com	194946	423690
7	419489	DESALEY AVROS	23 COOL STREET	d.avros@aol.com	957234	330795
8	627963	GUY RAYMOND	22 ROYHILL CLOSE	g.raymond@hotmail.com	202885	149333
9	515022	JESSE BEANE	85 DEVALINE LANE	j.beane@hotmail.com	532993	245172
10	186601	JOY FLOTTS	9 DEFILO STREET	j.flotts@gmail.com	402069	898131

IV: Testing

The following section focuses on three sets of tests that were conducted to ensure the validity of the project. This was completed by testing a given 3 of 6 tests and experimenting on the provided outcome(s). Find below a breakdown of the preset tests, including the assumed outcome and the official outcome.

Test 1 : Employment details of a particular student

Test Description	**SQL Query**
The following test focuses on printing the details of student(s) with their pre-existing employment details	***SELECT*** *gradrole, studentname, employmenttype* ***FROM*** *student1, employmentdetails1* ***WHERE*** *student1.studentid = employmentdetails1.studentid*
Expected Outcome	
Based on our expectations, we presume that the query should print the student(s) name, their employment detail(s) including whether the role is a graduate role.	
Actual Outcome	
The query printed our expected outcome.	

Result

Query Editor | Query History

```
1  select gradrole, studentname, employmenttype from student1, employmentdetails1
2  where student1.studentid = employmentdetails1.studentid
```

Data Output | Explain | Messages | Notifications

	gradrole boolean	studentname character varying (255)	employmenttype character varying (255)
1	true	SAMUEL JONES	FULL-TIME
2	false	ANNA FELICITY	PART-TIME
3	true	TOM JACKSON	FIXED
4	true	SAMUEL L JACKSON	FULL-TIME
5	false	JAMES BOND	PART-TIME
6	true	JEFF SHREEVES	FIXED
7	false	ALAN SMITH	FULL-TIME
8	false	CHRIS KAMARA	FIXED
9	true	MARTIN TYLER	PART-TIME
10	false	EMILE HESKEY	PART-TIME

Test 2: Feedback from employed/unemployed students

Test Description	**SQL Query**
The following test was designed to focus on collecting feedback from students who are employed/unemployed.	**SELECT** student1.studentid, studentname, employmenttype,feedback **FROM** student1, programme1, employmentdetails1 **WHERE** student1.studentid = employmentdetails1.studentid **AND** student1.studentid = programme1.studentid
Expected Outcome	
The expected outcome for this test was to display the feedback from students with their corresponding existing employment status.	
Actual Outcome	
The actual outcome resulted in the student details and feedback being printed, including employment status, meeting our expectations.	

Result

Query Editor Query History

```
1  select student1.studentid, studentname, employmenttype,feedback from student1, programme1, employmentdetails1
2  where student1.studentid = employmentdetails1.studentid
3  and student1.studentid = programme1.studentid
```

Data Output Explain Messages Notifications

	studentid integer	studentname character varying (255)	employmenttype character varying (255)	feedback character varying (255)
1	29409552	SAMUEL JONES	FULL-TIME	GOOD, BUT THE SUPPORT FOR THIS PROGRAMME COULD BE BETTER.
2	26347199	ANNA FELICITY	PART-TIME	SOME OF THE MODULES WERE OK, BUT THE OTHERS WERE NOT AS GOOD.
3	27673198	TOM JACKSON	FIXED	FOUND IT TOO EASY, THE SUPPORT WAS PRETTY GOOD.
4	12891283	SAMUEL L JACKSON	FULL-TIME	WAS OK, BUT COULD HAVE BEEN BETTER.
5	43739173	JAMES BOND	PART-TIME	REALLY ENJOYED IT, WOULD DO IT AGAIN.
6	28281711	JEFF SHREEVES	FIXED	IT WAS INTERESTING HOW THE MARKETS IN THE WORLD WORKED.
7	11182212	ALAN SMITH	FULL-TIME	ALLOWED FOR THE STUDENTS TO COME UP WITH SOME CREATIVITY FOR SOLVING PARTICULAR PROBLEMS.
8	18187221	CHRIS KAMARA	FIXED	WASN'T THAT GOOD, THE LECTURER HADN'T SET OUT THE WORK NICELY.
9	98287112	MARTIN TYLER	PART-TIME	VERY GOOD, REALLY LEARNT ALOT FROM SUPPORT.
10	69696969	EMILE HESKEY	PART-TIME	IT WAS OK, BUT I FELT THERE COULD HAVE BEEN MORE CHANCES FOR PRACTICALS.

Test 3: Average grade of the students that are currently employed/unemployed

Test Description	**SQL Query**
The 3rd test focuses on finding the average grade of student(s) that are employed/unemployed.	**SELECT** avr(finalpercentage) :: numeric(10,1) **FROM** student1
Expected Outcome	
The expected outcome was to present the student(s) details and their average grade.	
Actual Outcome	
As illustrated, the average grade was printed but only 1 set. This was due to the preset inserted data focused on only having 1 individual completing the course and having a generated average.	

Result

```
1  select avg(finalpercentage) :: numeric(10,1) from student1
```

Data Output | Explain | Messages | Notifications

	avg numeric (10,1)
1	88.2

References

[1]: https://www.studytonight.com/dbms/second-normal-form.php
[2]: https://www.studytonight.com/dbms/third-normal-form.php
https://www.w3schools.com/sql/sql_datatypes.asp
https://www.journaldev.com/16774/sql-data-types
https://www.sqlservertutorial.net/sql-server-basics/sql-server-inner-join/