

# DataBase Management Project

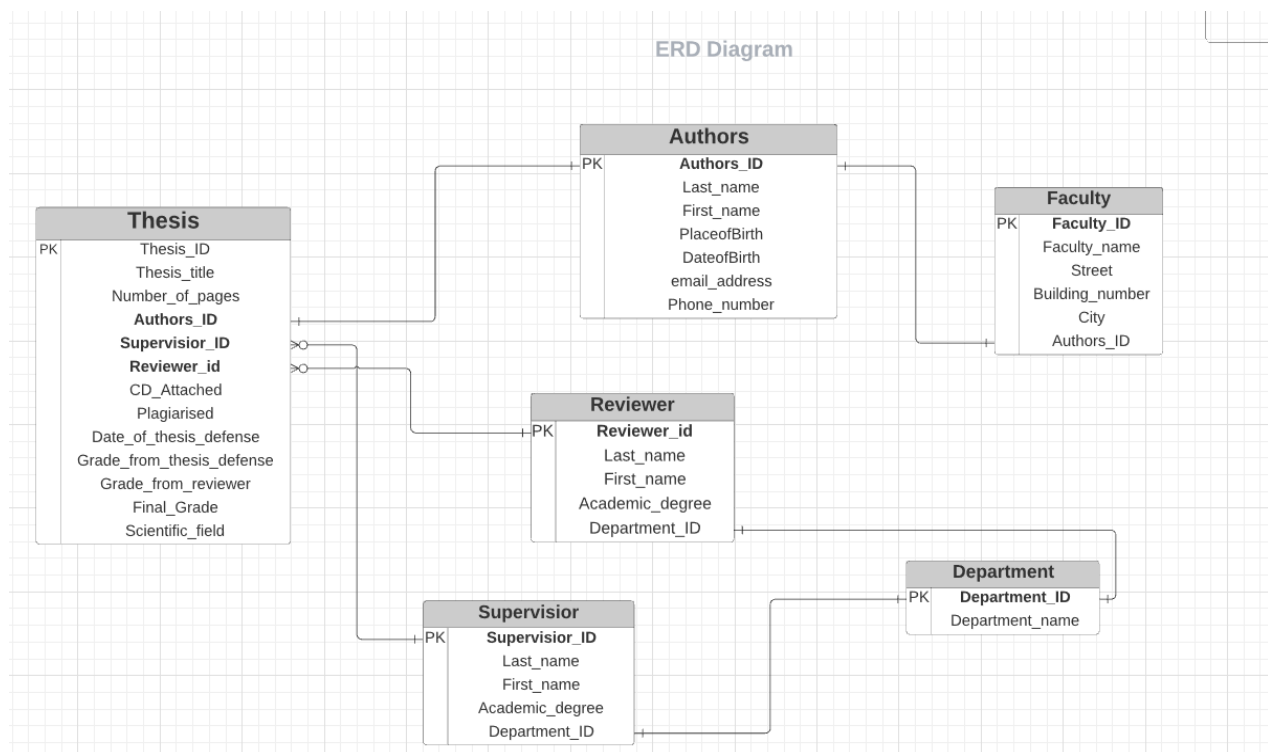
## Introduction:

Database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling technique. And DMBS which means DataBase Management System is one of the most important features of database. The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data.

Modern DBs have built in tools ensuring adequate access, manipulation and update of data collected in a computer system. The most important characteristics of DBMS includes : operating on large and very large datasets, managing complex data structures, long-term operation.

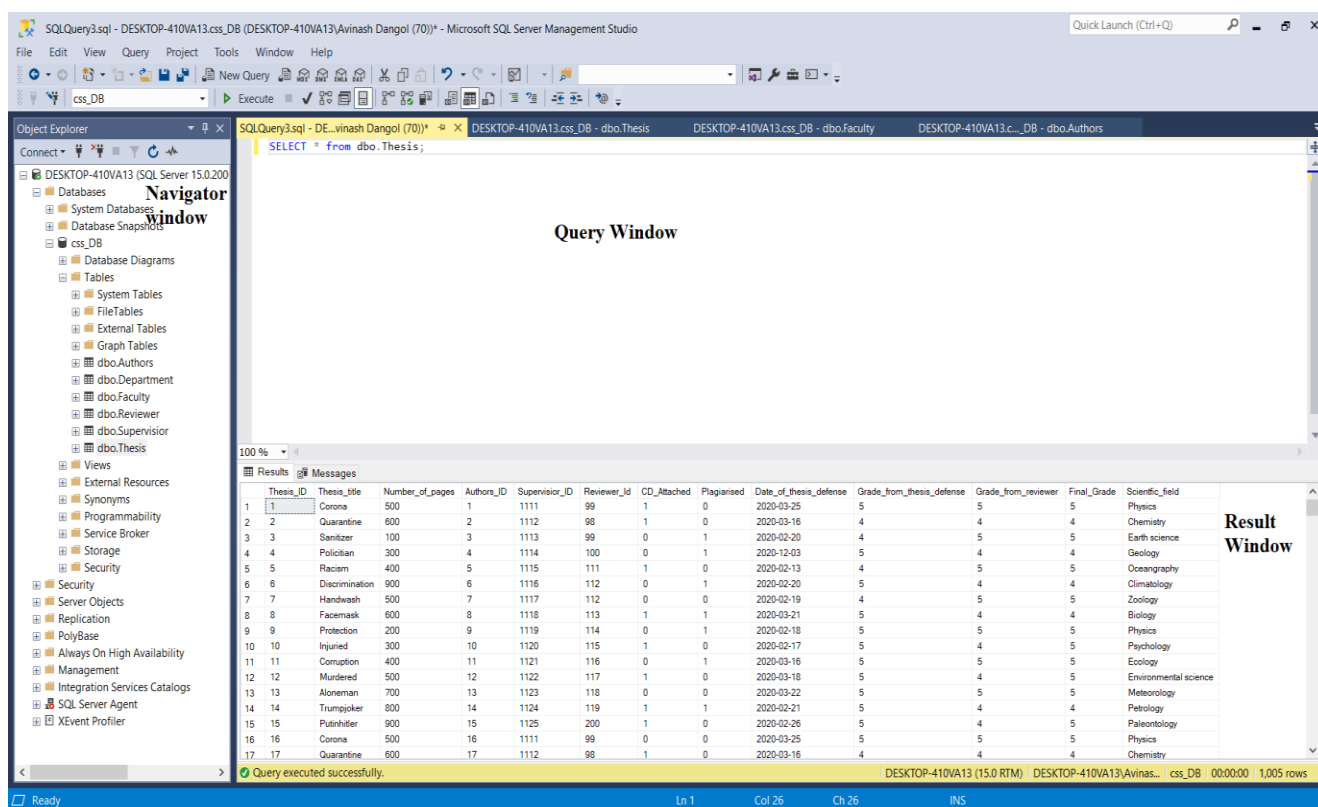
There are many Database Management Systems available in the market. The most popular ones are: Oracle, MS SQL Server, DB2, Sybase, Informix, Adabase, ObjectStore, MS Access. So, we have used MS SQL server studio 2018 to create my database that is CSS\_DataBase(css\_DB) of Diploma Thesis.

## ERD (Entity Relationship Diagram) of Diploma Thesis



So, First of all we created the ERD diagram which is also called Entity Relationship Diagram by using online software called Lucidchart. Where we have made the tables, database owner of Thesis, Authors, Reviewer, Supervisor, Department, Faculty where Thesis is a dbo(database owner) in which Thesis\_ID is primary key and thesis\_title, Number\_of\_pages, Authors\_ID, Supervisor\_ID, Reviewer\_id, CD\_attached, Plagiarised, Date\_of\_thesis\_defense, Grade\_from\_thesis\_defense, Grade\_from\_reviewer\_final\_grade, Scientific\_field are the foreign key and same as for other dbo as well.

**DBMS Interfaces:** A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself. A DBMS interface could be a web client, a local client that runs on a desktop computer, or even a mobile app.



**DBMS ensures ACID properties:**

- Atomicity: all-or-nothing property
- Consistency: a transaction brings the database from one consistent state to another
- Isolation: concurrent transactions appear to run in isolation
- Durability: DBMS ensures durability of transactions

## Data Normalization:

It is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. Normalization is a database design technique, which is used to design a relational database table up to higher normal form. The process is progressive, and a higher level of database normalization cannot be achieved unless the previous levels have been satisfied. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.

**Normal forms** are used to eliminate or reduce redundancy in database tables.

**First Normal Form:** A database is in first normal form if it satisfies the following conditions:

- Contains only atomic values
- There are no repeating groups

. A relation is in first normal form if every attribute in that relation is singled valued attribute.

Example: Relation STUDENT in table is in first normal form is satisfied, as the columns on each table all hold just one value.

Authors_ID	Last_name	First_name	PlaceofBirth	DateofBirth	email_addr...	Phone_num...
1	Dangol	Avinash	Kathmandu	1995-03-28	avi@gmail.c...	739403070
2	Puri	Laky	Malta	1996-03-25	laky@gmail....	739403030
3	Fiak	Firat	Tokyo	1990-12-25	firat@gmail...	739506020
4	Maharjan	Manoj	Jhor	1996-11-11	mag@gmai...	739562389
5	Kami	Nikita	Warsaw	1997-06-14	kami@gmai...	885269875
6	Magar	Sabita	Paris	1992-01-16	sabi@gmail...	739506010
7	Rana	Sami	Berlin	1996-09-25	sami@gmai...	759105268
8	Shrestha	Prawjol	Narobi	1993-10-10	praw@gmai...	579108158
9	Tamang	Ashish	Mombai	1994-06-16	ashish@gm...	579158236
10	Lama	Anju	China	1992-09-19	lama@gmai...	579865232

**Second Normal Form:** A database is in second normal form if it satisfies the following conditions:

- It is in first normal form
- All non-key attributes are fully functional dependent on the primary key

In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

Faculty_ID	Faculty_na...	Street	Building_n...	City	Authors_ID
9090	Computer s...	ul.jasielska	86	warsaw	1
9091	Business stu...	ul.Zmk	7	kathmandu	2
9092	Hotel mana...	ul.rando	12	patan	3
9093	Tourism	ul.onz	15	jalamfar	4
9094	Electrical	ul.tarasy	18	istambal	5
9095	IBM	ul.turi	19	bali	6
9096	Economical	ul.lachhi	20	thailand	7
9097	Architecture	ul.karem	25	paris	8
9098	Civil	ul.garem	23	oslo	9
9099	humanities	ul.ochota	28	venecca	10
9100	Nursing	ul.krakoska	66	lodz	11

The table given above is in the first normal form where Faculty\_ID is a primary key.

Authors_ID	Last_name	First_name	PlaceofBirth	DateofBirth	email_addr...	Phone_num...
1	Dangol	Avinash	Kathmandu	1995-03-28	avi@gmail.c...	739403070
2	Puri	Laky	Malta	1996-03-25	laky@gmail....	739403030
3	Fiak	Firat	Tokyo	1990-12-25	fiat@gmail...	739506020
4	Maharjan	Manoj	Jhor	1996-11-11	mag@gmai...	739562389
5	Kami	Nikita	Warsaw	1997-06-14	kami@gmai...	885269875
6	Magar	Sabita	Paris	1992-01-16	sabi@gmail...	739506010
7	Rana	Sami	Berlin	1996-09-25	sami@gmai...	759105268
8	Shrestha	Prawjol	Narobi	1993-10-10	praw@gmai...	579108158
9	Tamang	Ashish	Mombai	1994-06-16	ashish@gm...	579158236
10	Lama	Anju	China	1992-09-19	lama@gmai...	579865232

And the above table is another first normal form where Authors\_ID is primary key.

Now, To bring this table to second normal form, we break the table into two tables, and now we have the following:

Faculty_ID	Authors_ID
9090	1
9091	2
9092	3
9093	4
9094	5
9095	6
9096	7
9097	8
9098	9

**Therefore, this table is second normalization form.**

**Third Normal Form:** A database is in third normal form if it satisfies the following conditions:

- It is in second normal form
- There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the table: A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

so, From the table below,

Thesis_ID	Thesis_title	Number_of...	Authors_ID	Supervisor...	Reviewer_Id
1	Corona	500	1	1111	99
2	Quarantine	600	2	1112	98
3	Sanitizer	100	3	1113	99
4	Policitian	300	4	1114	100
5	Racism	400	5	1115	111
6	Discriminati...	900	6	1116	112
7	Handwash	500	7	1117	112
8	Facemask	600	8	1118	113
9	Protection	200	9	1119	114
10	Injured	300	10	1120	115
11	Corruption	400	11	1121	116

In the table , [Thesis\_ID] determines [Thesis\_title], and [Thesis\_title] determines [Authors\_ID]. Therefore, [Thesis\_ID] determines [Authors\_ID] via [Thesis\_title] and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two as follows:

**Table\_Thesis**

Thesis_ID	Authors_ID	Number_of...
1	1	500
2	2	600
3	3	100
4	4	300
5	5	400
6	6	900
7	7	500
8	8	600

**Table\_Authors**

Thesis_title	Authors_ID
Corona	1
Quarantine	2
Sanitizer	3
Policitian	4
Racism	5
Discriminati...	6
Handwash	7
Facemask	8

Now all non-key attributes are fully functional dependent only on the primary key. In [TABLE\_Thesis], both [Authors\_ID] and [Number\_of\_pages] are only dependent on [Thesis\_ID]. In [TABLE\_Authors], [Thesis\_title] is only dependent on [Authors\_ID].

**Database query:** A database query is a request for data from a database. Usually the request is to retrieve data; however, data can also be manipulated using queries. The data can come from one or more tables, or even other queries.

**Transactions:** A transaction, in the context of a database, is a logical unit that is independently executed for data retrieval or updates. In relational databases, database transactions must be atomic, consistent, isolated and durable--summarized as the ACID acronym. One example is a transfer from one bank account to another: the complete transaction requires subtracting the amount to be transferred from one account and adding that same amount to the other.

Transactions are completed by COMMIT or ROLLBACK SQL statements, which indicate a transaction's beginning or end. The ACID acronym defines the properties of a database transaction, as follows:

- Atomicity: A transaction must be fully complete, saved (committed) or completely undone (rolled back). A sale in a retail store database illustrates a scenario which explains atomicity, e.g., the sale consists of an inventory reduction and a record of incoming cash. Both either happen together or do not happen - it's all or nothing.
- Consistency: The transaction must be fully compliant with the state of the database as it was prior to the transaction. In other words, the transaction cannot break the database's constraints. For example, if a database table's Phone Number column can only contain numerals, then consistency dictates that any transaction attempting to enter an alphabetical letter may not commit.
- Isolation: Transaction data must not be available to other transactions until the original transaction is committed or rolled back.
- Durability: Transaction data changes must be available, even in the event of database failure.

Example of transactions:

```
1)
BEGIN TRANSACTION
BEGIN TRY
UPDATE dbo.Authors SET PlaceofBirth =('Kathmandu') Where [Authors_ID] = 1
INSERT INTO dbo.Authors (Last_name) VALUES ('Dangol')
COMMIT TRANSACTION;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION
END CATCH
SELECT * FROM Authors
```

Results:

we have modified the transaction where we updated the placeofbirth as Kathmandu in Authors\_ID of dbo.Authors table. SO as we can see in table there is the change in Authors\_ID number 2. And we used a ROLLBACK TRANSACTION statement to rollback the transaction meaning that none of the data actually changed.

The screenshot shows a SQL Server Enterprise Manager window with the following tabs: SQLQuery2.sql - DE...vinash Dangol (60)\*, SQLQuery1.sql - DE...vinash Dangol (59)\*, DESKTOP-410VA13.cs... - dbo.Supervisor\*, and DESKTOP-410VA13...DB - dbo.Reviewer\*.

The SQL query executed is:

```

BEGIN TRANSACTION
BEGIN TRY
UPDATE dbo.Authors SET PlaceofBirth = ('Kathmandu') Where [Authors_ID] = 2
INSERT INTO dbo.Authors (Last_name) VALUES ('Dangol')
COMMIT TRANSACTION;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION
END CATCH
Select * from Authors

```

The Results pane shows the following data:

Authors_ID	Last_name	First_name	PlaceofBirth	DatedOfBirth	email_address	Phone_number
1	Dangol	Avinash	Kathmandu	1995-03-28	avi@gmail.com	739403070
2	Puri	Laky	Kathmandu	1996-03-25	laky@gmail.com	739403030
3	Fiak	Firat	Tokyo	1990-12-25	fiat@gmail.com	739506020
4	Maharjan	Manoj	Jhor	1996-11-11	mag@gmail.com	739562389
5	Kami	Nikita	Warsaw	1997-06-14	kami@gmail.com	885269875
6	Magar	Sabita	Paris	1992-01-16	sabi@gmail.com	739506010
7	Rana	Sami	Berlin	1996-09-25	sami@gmail.com	759105268
8	Shrestha	Prawjol	Narobi	1993-10-10	praw@gmail.com	579108158
9	Tamang	Ashish	Mombai	1994-06-16	ashish@gmail.com	579158236
10	Lama	Anju	China	1992-09-19	lama@gmail.com	579865232
11	Regmi	Damodar	NewDelhi	1998-05-20	regmi@gmail.com	579868532
12	Damai	Mukul	Vietnam	1993-12-06	damai@gmail.com	579653228
13	Sarki	Billgates	Jumla	1991-03-22	bill@gmail.com	739403060
14	Maharjan	Mark	Dang	1999-07-25	mark@gmail.com	88895632
15	Kc	Hemant	Soul	1998-12-02	kc1@gmail.com	888256983
16	Dangol	Avinash	Kathmandu	1995-03-28	avi@gmail.com	739403070
17	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030

The status bar at the bottom indicates: Query executed successfully. DESKTOP-410VA13 (15.0 RTM) DESKTOP-410VA13\Avinas... css\_DB 00:00:00 1,010 rows

2)

```

BEGIN TRANSACTION
UPDATE Thesis SET Grade_from_thesis_defense = '4'
WHERE Grade_from_thesis_defense = '5'

SAVE TRANSACTION A
DELETE Thesis WHERE Thesis_ID = 5
ROLLBACK TRANSACTION A
COMMIT

SELECT * FROM Thesis

```

## Results:

The screenshot shows a SQL IDE with three tabs: SQLQuery3.sql, SQLQuery2.sql, and SQLQuery1.sql. The active tab, SQLQuery1.sql, contains the following SQL code:

```
BEGIN TRANSACTION
UPDATE Thesis SET Grade_from_thesis_defense = '4'
WHERE Grade_from_thesis_defense = '5'

SAVE TRANSACTION A
DELETE Thesis WHERE Thesis_ID = 5
ROLLBACK TRANSACTION A
COMMIT
SELECT * FROM Thesis
```

Below the code editor, the 'Results' pane displays a table with 13 columns: Thesis\_ID, Thesis\_title, Number\_of\_pages, Authors\_ID, Supervisor\_ID, Reviewer\_ID, CD\_Attached, Plagiarised, Date\_of\_thesis\_defense, Grade\_from\_thesis\_defense, Grade\_from\_reviewer, Final\_Grade, and Scientific\_field. The table contains 17 rows of data. The status bar at the bottom indicates 'Query executed successfully.' and '1,005 rows'.

Thesis_ID	Thesis_title	Number_of_pages	Authors_ID	Supervisor_ID	Reviewer_ID	CD_Attached	Plagiarised	Date_of_thesis_defense	Grade_from_thesis_defense	Grade_from_reviewer	Final_Grade	Scientific_field
1	Corona	500	1	1111	99	1	0	2020-03-25	4	5	5	Physics
2	Quarantine	600	2	1112	98	1	0	2020-03-16	4	4	4	Chemistry
3	Sanitizer	100	3	1113	99	0	1	2020-02-20	4	5	5	Earth science
4	Politician	300	4	1114	100	0	1	2020-12-03	4	4	4	Geology
5	Racism	400	5	1115	111	1	0	2020-02-13	4	5	5	Oceanography
6	Discrimination	900	6	1116	112	0	1	2020-02-20	4	4	4	Climatology
7	Handwash	500	7	1117	112	0	0	2020-02-19	4	5	5	Zoology
8	Facemask	600	8	1118	113	1	1	2020-03-21	4	4	4	Biology
9	Protection	200	9	1119	114	0	1	2020-02-18	4	5	5	Physics
10	Injured	300	10	1120	115	1	0	2020-02-17	4	4	5	Psychology
11	Corruption	400	11	1121	116	0	1	2020-03-16	4	5	5	Ecology
12	Murdered	500	12	1122	117	1	0	2020-03-18	4	4	5	Environmental science
13	Aloneman	700	13	1123	118	0	0	2020-03-22	4	5	5	Metereology
14	Trumpjoker	800	14	1124	119	1	1	2020-02-21	4	4	4	Petrology
15	Pulitshiller	900	15	1125	200	1	0	2020-02-26	4	4	5	Paleontology
16	Corona	500	16	1111	99	0	0	2020-03-25	4	5	5	Physics
17	Quarantine	600	17	1112	98	1	0	2020-03-16	4	4	4	Chemistry

In this table we have modified the transaction as begin transaction where we have updated thesis and set Grade\_from\_thesis\_defense as 4, it will replace all the values of column in Grade\_from\_thesis\_defense. As we can see in the table Grade\_from\_thesis\_defense is 4. And then we issued a ROLLBACK TRANSACTION statement to rollback the transaction meaning that none of the data actually changed.

**Procedures:** Procedures (sometimes referred to as Stored Procedures or Procs) are subroutines that can contain one or more SQL statements that perform a specific task. They can be used for data validation, access control, or to reduce network traffic between clients and the DBMS servers.

Examples of procedures :

- 1.(a) 

```
CREATE PROC What_DB_is_this
AS
SELECT DB_NAME () AS THISDB;

EXEC What_DB_is_this
```
- (b) 

```
CREATE PROC What_DB_is_that
AS
SELECT DB_NAME () AS THATDB;

EXEC What_DB_is_that
```



## Results:

This the procedure created to find out the actual name of the database. In query (a) We wrote query to find out what db this belongs to and another query in (b) we had written what db that belongs to . This and That it executes to same database as we created. And it shows css\_DB as it is my database.

2).

```
CREATE PROCEDURE Reviewer_ID
@deg nvarchar(40)= PHD
AS
SELECT Last_name
FROM dbo.Reviewer
WHERE [Academic_degree]=@deg
```

## Results:

[deg: a **database** of essential genes]

## Views:

A database view is a searchable object in a database that is defined by a query. Though a view doesn't store data, some refer to a views as “virtual tables,” you can query a view like you can a table. A view can combine data from two or more table, using joins, and also just contain a subset of information.

According to the views from table(MS SMS):

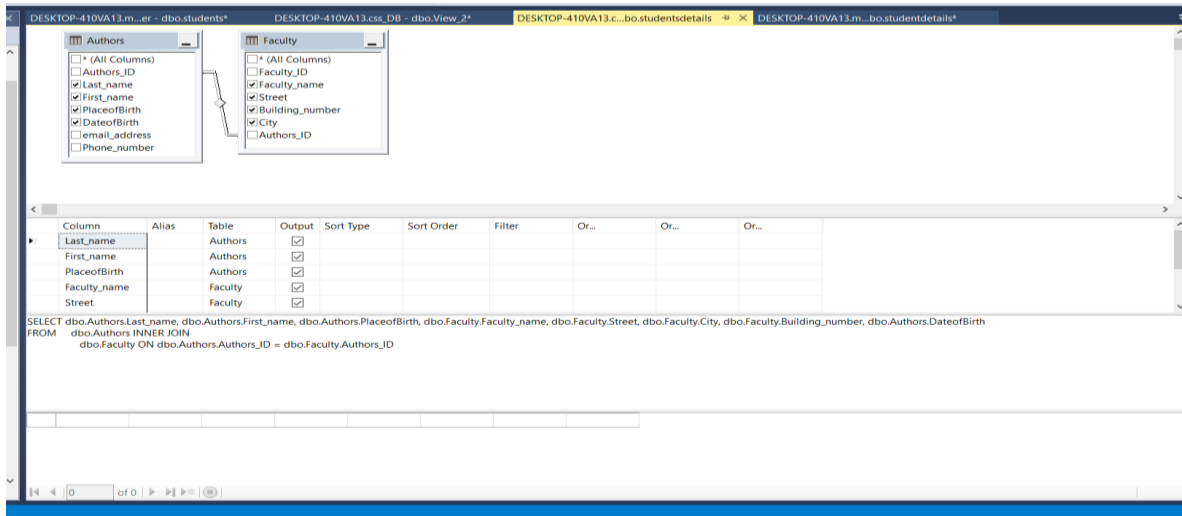
a) dbo.studentdetails

```
SELECT dbo.Authors.Last_name, dbo.Authors.First_name, dbo.Authors.PlaceofBirth,
dbo.Faculty.Faculty_name, dbo.Faculty.Street, dbo.Faculty.City, dbo.Faculty.Building_number,
dbo.Authors.DateofBirth
```

```
FROM    dbo.Authors INNER JOIN
```

```
        dbo.Faculty ON dbo.Authors.Authors_ID = dbo.Faculty.Authors_ID
```

DESIGN:



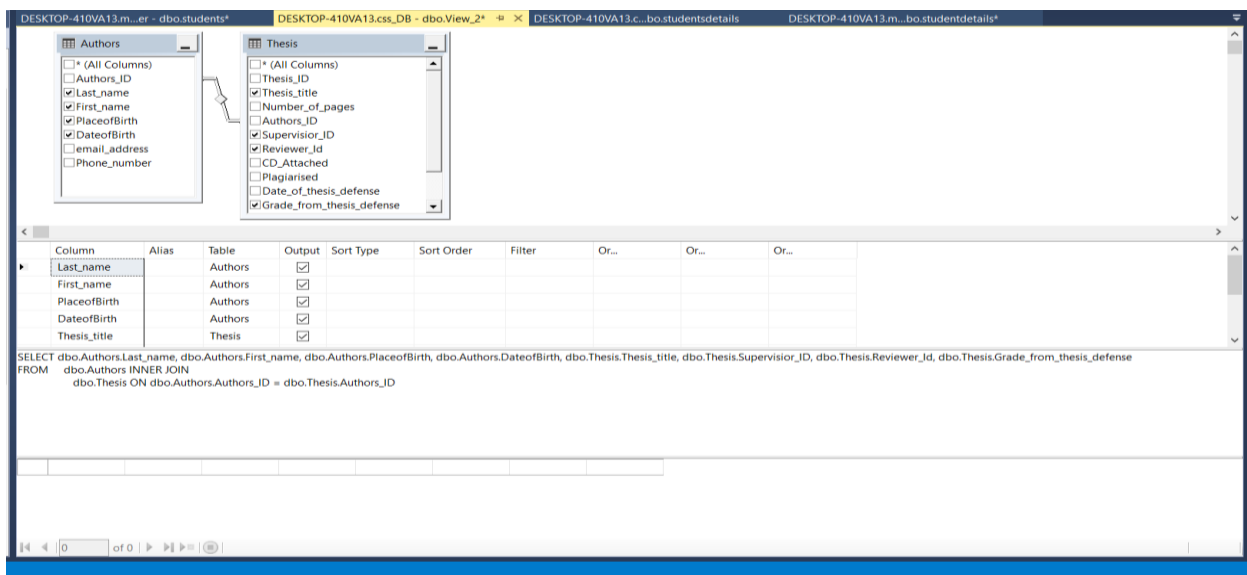
b)dbo.students

SELECT dbo.Authors.Last\_name, dbo.Authors.First\_name, dbo.Authors.PlaceofBirth,  
dbo.Authors.DateofBirth, dbo.Thesis.Thesis\_title, dbo.Thesis.Supervisor\_ID,  
dbo.Thesis.Reviewer\_Id, dbo.Thesis.Grade\_from\_thesis\_defense

FROM    dbo.Authors INNER JOIN

      dbo.Thesis ON dbo.Authors.Authors\_ID = dbo.Thesis.Authors\_ID

DESIGN:



**Triggers:** A triggers is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

### BEFORE and AFTER of Trigger:

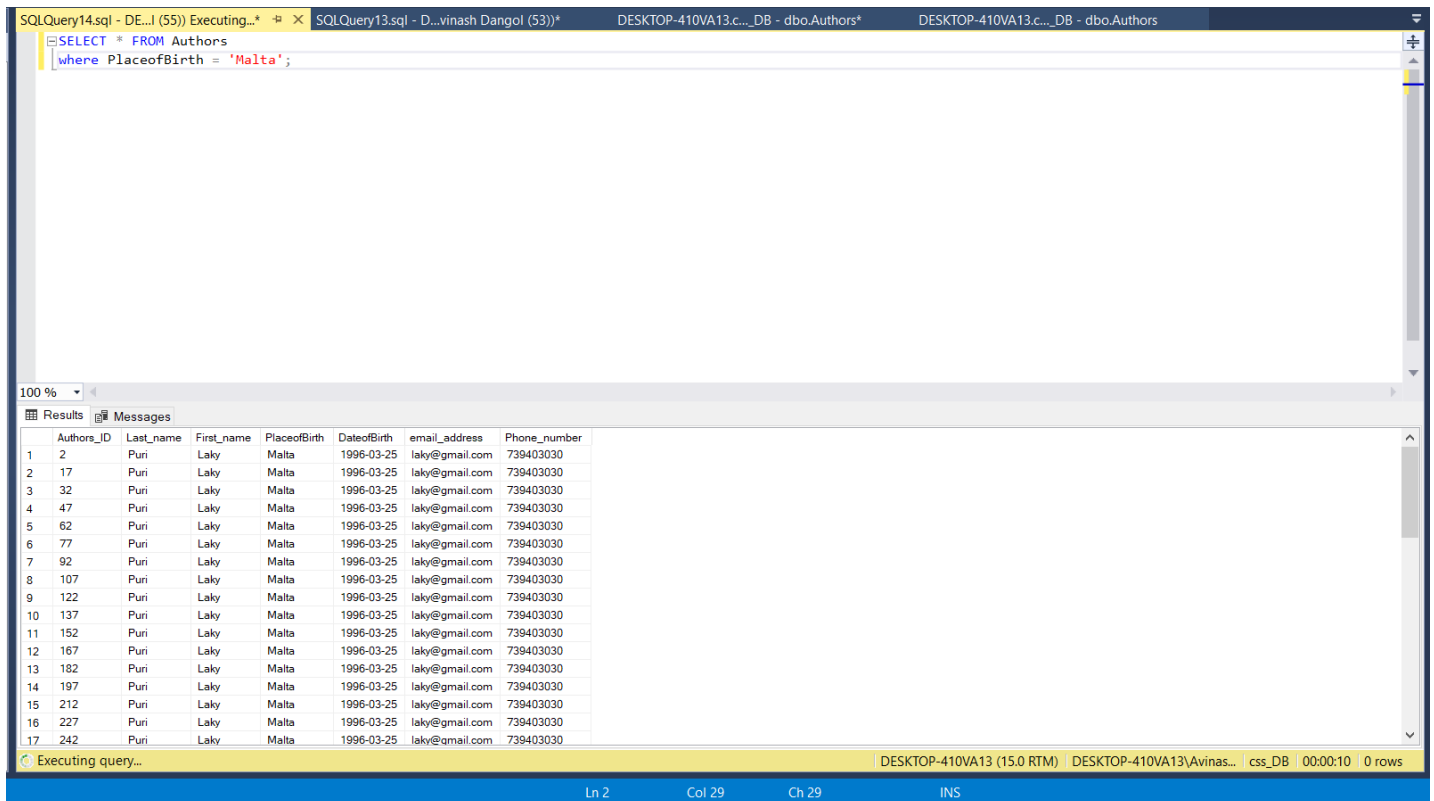
BEFORE triggers run the trigger action before the triggering statement is run.

AFTER triggers run the trigger action after the triggering statement is run.

## Indexes:

A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Some databases extend the power of indexing by letting developers create indexes on functions or expressions.

### Index utilization:



SQLQuery14.sql - DE... (55) Executing... SQLQuery13.sql - D...vinash Dangol (53)\* DESKTOP-410VA13.c...DB - dbo.Authors\* DESKTOP-410VA13.c...DB - dbo.Authors

```
SELECT * FROM Authors
where PlaceOfBirth = 'Malta';
```

100 %

Results Messages

	Authors_ID	Last_name	First_name	PlaceOfBirth	DateOfBirth	email_address	Phone_number
1	2	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
2	17	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
3	32	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
4	47	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
5	62	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
6	77	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
7	92	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
8	107	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
9	122	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
10	137	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
11	152	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
12	167	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
13	182	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
14	197	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
15	212	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
16	227	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030
17	242	Puri	Laky	Malta	1996-03-25	laky@gmail.com	739403030

Executing query... DESKTOP-410VA13 (15.0 RTM) DESKTOP-410VA13\Avinas... css\_DB 00:00:10 0 rows

Ln 2 Col 29 Ch 29 INS

\*\*\*\*\*Thankyou\*\*\*\*\*

**Name : Avinash Dangol**

**Major : Computer science**

**Semester : 4<sup>th</sup> semester**

**Student ID : 99070**