University of Salford MANCHESTER

Advanced Databases Assignment

2021-22

Sophie Mannion

@00641004

Contents

Task 1: Young Lives investigation into child poverty 4

1. Introduction 4
2. Design Rationale 5
3. Relational Schema 6
4. Design Considerations 7
5. T-SQL statements 12
6. Database security 20
7. Database backup 20
8. Conclusion 20

Task 2: School Survey Vietnam 2016-17 21

1. Introduction 21
2. Design Rationale 22
3. Design Considerations 24
4. T-SQL statements 28
5. Report Design 35
6. Database security 39
7. Database backup 39
8. Conclusion 39

Task 3: Crime profiler 40

1. Introduction 40
2. Design Rationale 40
3. Design Considerations 41
4. T-SQL statements 46
5. Report Design 50
6. Database security 54
7. Database backup 55
8. Data Science/Business Intelligence Techniques 55

9. Conclusion 55

References 56

Task 1: Young Lives investigation into child poverty

Databases offer the opportunity to ensure an organisation's data is maintained through protecting data integrity, data concurrency and security. The storage of data in one central location can allow multiple users to access required data to meet organisation objectives and goals efficiently and quickly. This is especially important with the amount of data generated in the present age. This task aims to import data from the Young Lives study into a database, creating logically related tables, with the aim of enhancing researchers' ability to investigate child poverty across Ethiopia, Peru, India, and Vietnam

1. Introduction

For a long time, data was stored independently in separate files. This approach has a lot of drawbacks which can make business use inefficient (Connolly and Begg, 2015). Linking related data kept in separate files required creating new files to store combined data. Data could often be duplicated across departments and teams. Any updating of data could require updating data across multiple files, which will likely create data inconsistencies and future errors. In the end, any novel user requirements of the data would have to be custom-written and produce by the IT team, increasing the amount of time business tasks could take. With the increasing amounts of data produced, this system would only become more difficult (Connolly and Begg, 2015).

The development of databases has overcome many of these drawbacks. A database is a collection of related data which meets the needs of the organisation (Connolly and Begg, 2015). They provide a centralised location for all relevant and useful data, in the same format, allowing users to efficiently access required data. As related data is linked within the database, users are able to access combinations of data from different tables as required without the need for custom files and programs to be created (Connolly and Begg, 2015). The Database Management System is the software which allows users to interact with the stored data. They also provide database security, protect data integrity, create a concurrency control system so multiple users can access the system at one time, and a recovery system which reverts the database back to a previous consistent state in the event of failure, preventing data loss (Connolly and Begg, 2015). SQL Server is a popular DBMS software developed by Microsoft, which offers all of these services. Overall, databases allow organisations to speed up their analysis and therefore decision making process, and allow organisations to collect more data than they previously would have been able to (Connolly and Begg, 2015, Elmasri and Navathe, 2017).

The Young Lives study is a longitudinal investigation into the effects of child poverty across Ethiopia, India, Peru and Vietnam (Crivello et al., 2013). Data was collected from a sample of 12,000 children from these four countries across five rounds (15 years in total). The children were surveyed every 3-4 years, with data collected on a variety of aspects of the child's life, from their education to their family to their home. The organisation would benefit from a database system to store collected data efficiently and build reports to analyse child poverty. This task will use SQL Server 2018 to import the recorded data into a database, organise the data into logically related tables and create example reports that users could use to analyse child poverty.

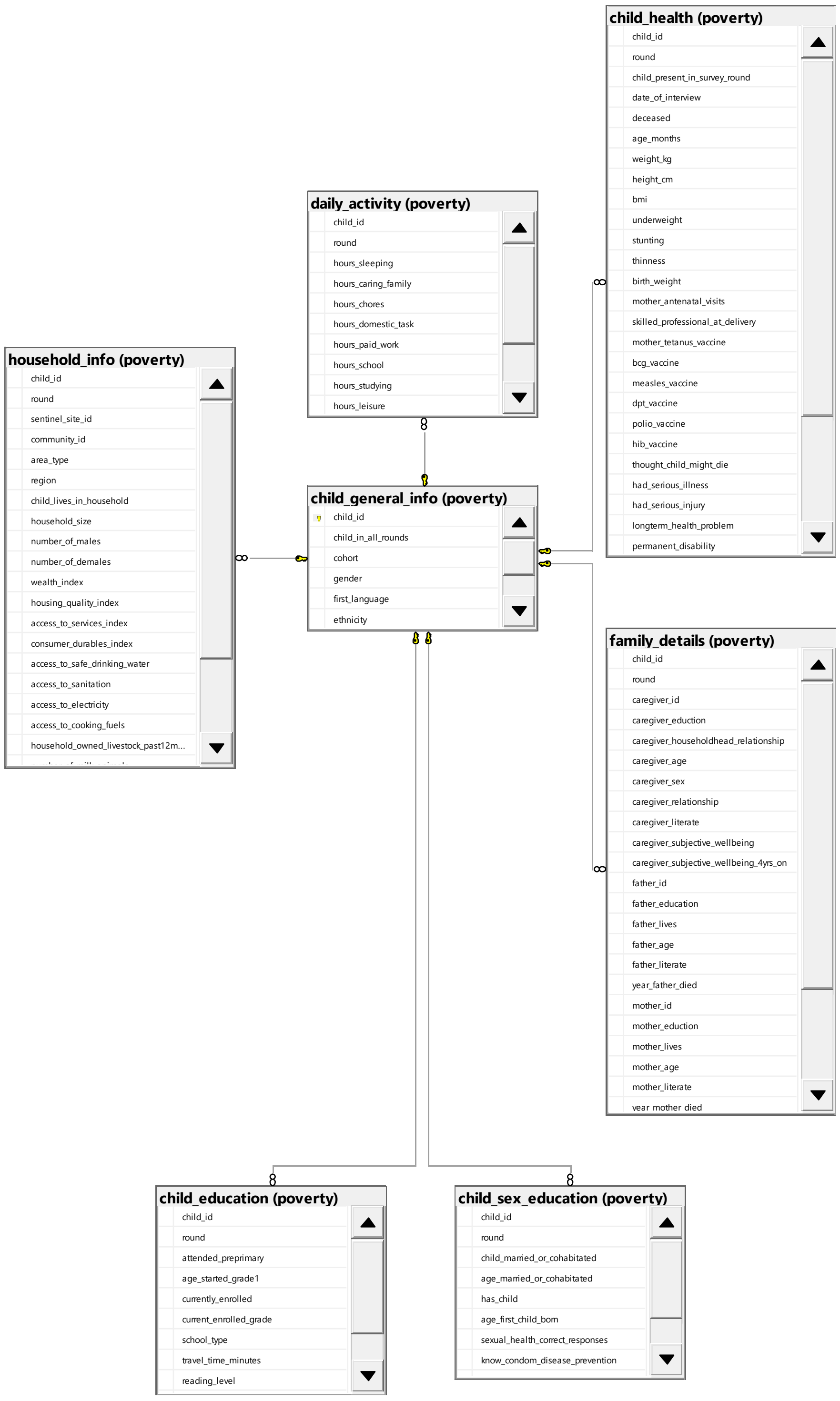
2. Design Rationale

SQL Server is an example of a Relational Database Management System, which is based on the relational data model proposed by Codd (Codd, 1972). All data is represented in tables (relations) with columns (attributes) and rows (tuple) which contain one value per attribute. The relational model must meet a set of objectives: allow data independence and allow for normalisation which deals with problems relating to data semantics, consistency, and redundancy. To ensure these objectives are met, relations and attributes must have a distinct name, attribute values must be the same type, tuples must be distinct, each attribute contains a single value per tuple, and the order of tuples and attributes must not be significant (Connolly and Begg, 2015). There are a number of design considerations which help to ensure these requirements are met, including normalisation of relations and setting constraints, which will be discussed in detail later (Elmasri and Navathe, 2017).

Before database implementation, conceptual design of the database is essential in order for the database to meet the goals of the users. Understanding the overall business goals and the users requirements of the database allows the selection of appropriate data and how this data should be organised within the database (Connolly and Begg, 2015). The Young Lives study has already gathered the required data to meet their goals. The aim of the required database is storing the longitudinal data appropriately for each country in a way which facilitates user investigation into child poverty across different countries and the effects of childhood poverty across an individual's life. For the conceptual design of the database, it is popular to follow the entity-relationship (ER) model. The first step is to identify entities and their attributes. Entities are 'objects' with their own independent existence and can be either physical or abstract (Connolly and Begg, 2015). Each entity then has corresponding attributes which describe the properties of interest to the users (Elmasri and Navathe, 2017). In this instance, each child can be considered to be an entity and the measurements taken for each round are considered to be the attributes of the child. Most of the attributes can be considered to be multi-valued attributes as they have multiple values recorded for each child according to the round. However, some are single-valued attributes as they are consistent through each round. Some questions were only asked to individuals belonging to a particular cohort or the child may not have been present for particular rounds (Briones, 2018). Any data missing data for each child needs to be filled in with a NULL value. However, one attribute must not have any NULL values and this is the key attribute (Connolly and Begg, 2015). The primary key attribute is distinct for each entity so can be used to identify different entities. In the Young Lives study, each child was given a unique identifier to distinguish their data throughout the rounds (Briones, 2018). This will be used as the key attribute. Often in databases there are multiple entities which relate to each other and so can be linked together via these relationships. However, in this instance there is only one entity, so this type of mapping is not necessary. In the Young Lives study, a variety of attributes were recorded for each child, with some only being relevant to specific individuals or groups and some remaining consistent throughout. It is still important to consider minimising redundancy within the database model (Connolly and Begg, 2015). Therefore, the relation can be broken down further, grouping attributes which are more similar into separate relations. Each relation will contain a foreign key which corresponds to the primary key (child's ID) so the data can be mapped to the corresponding child across tables.

3. Relational Schema

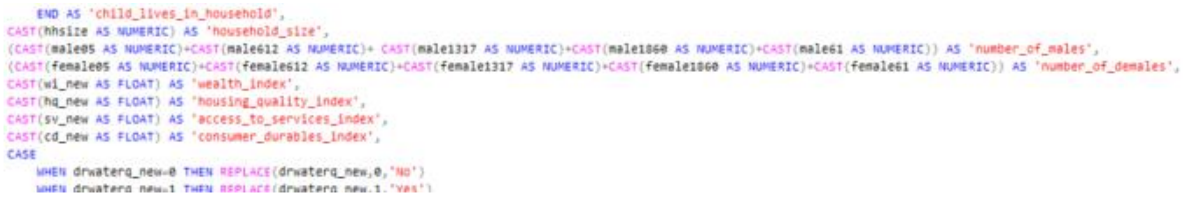
A relational database is often made up of multiple relations (tables). There are often relationships within relations and between different relations, which is referred to as a relational schema (Connolly and Begg, 2015). In this instance, attributes relating to the study were grouped by similarity in order to reduce redundancy and the number of NULL values in a table. The figure below is the schema diagram which represents the separate groups of attributes (tables) and how they are related. The main table contains the child's general information, which is consistent across rounds, such as their gender, ethnicity, religion, etc. The child's ID is the primary key for this table as each child's information is only recorded once. Other attributes were grouped according to the information they contained, such as details of family ('family_details'), household information ('household_info'), the child's health ('child_health'), the child's education ('child_education') and the child's sex education ('child_sex_education'). These are all linked to the main table which contains all the child's general information through the foreign key 'child_id'.



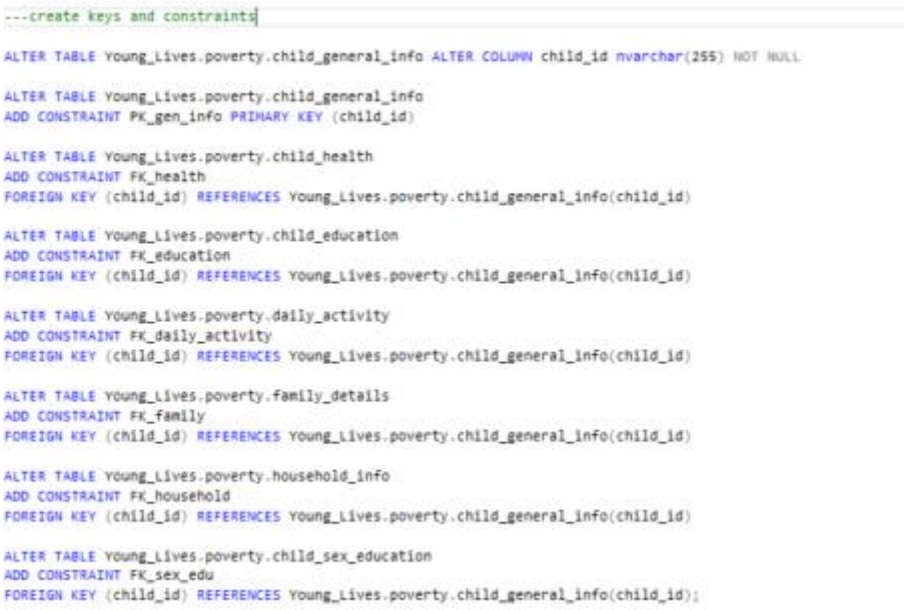
4. Design Considerations

The next stage is to consider the logical database design. This translates the conceptual model into a logical model where the model structure and ability to manage user requirements are evaluated. The stages of developing a logical data model include, normalisation, creating integrity constraints, check transactions can be handled by the model and that the model is secure (Connolly and Begg, 2015).

Constraints are necessary to consider in order to ensure data integrity is maintained. These are rules which prevent altering of information in the database which would impact the accuracy or quality of information stored. Constraints can be domain constraints, entity integrity constraints or referential integrity constraints (Elmasri and Navathe, 2017). Domain constraints involve setting column data types. In this instance, columns such as the number of family members in household (‘household_size’) were cast as numeric and columns such as wealth index (‘wi_new) were cast as decimals to ensure maintenance of decimal places. This ensures that strings cannot be inputted into these columns without causing an error. Entity integrity constraints involve preventing primary key values from being null, which in this instance is the child’s ID. Referential integrity constraints involve all foreign keys in a table matching up to the primary keys in another table. Additionally, running queries can sometimes produce errors which prevent them from continuing to run or produce unexpected results (Mukherjee, 2019). To raise awareness for any errors occurring, many sections of code include ‘TRY… CATCH’ statements which will print error messages should an error occur, allowing administrator intervention.



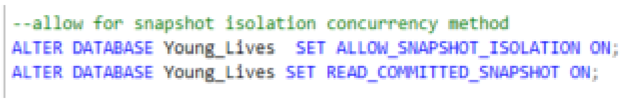
```
    END AS 'child_lives_in_household',
CAST(hhsize AS NUMERIC) AS 'household_size',
(CAST(male05 AS NUMERIC)+CAST(male612 AS NUMERIC)+ CAST(male1317 AS NUMERIC)+CAST(male1860 AS NUMERIC)+CAST(male61 AS NUMERIC)) AS 'number_of_males',
(CAST(female05 AS NUMERIC)+CAST(female612 AS NUMERIC)+CAST(female1317 AS NUMERIC)+CAST(female1860 AS NUMERIC)+CAST(female61 AS NUMERIC)) AS 'number_of_demales',
CAST(wi_new AS FLOAT) AS 'wealth_index',
CAST(hq_new AS FLOAT) AS 'housing_quality_index',
CAST(sv_new AS FLOAT) AS 'access_to_services_index',
CAST(cd_new AS FLOAT) AS 'consumer_durables_index',
CASE
    WHEN drwaterq_new=0 THEN REPLACE(drwaterq_new,0,'No')
    WHEN drwaterq_new=1 THEN REPLACE(drwaterq_new,1,'Yes')
```



```
---create keys and constraints

ALTER TABLE Young_Lives.poverty.child_general_info ALTER COLUMN child_id nvarchar(255) NOT NULL

ALTER TABLE Young_Lives.poverty.child_general_info
ADD CONSTRAINT PK_gen_info PRIMARY KEY (child_id)

ALTER TABLE Young_Lives.poverty.child_health
ADD CONSTRAINT FK_health
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id)

ALTER TABLE Young_Lives.poverty.child_education
ADD CONSTRAINT FK_education
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id)

ALTER TABLE Young_Lives.poverty.daily_activity
ADD CONSTRAINT FK_daily_activity
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id)

ALTER TABLE Young_Lives.poverty.family_details
ADD CONSTRAINT FK_family
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id)

ALTER TABLE Young_Lives.poverty.household_info
ADD CONSTRAINT FK_household
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id)

ALTER TABLE Young_Lives.poverty.child_sex_education
ADD CONSTRAINT FK_sex_edu
FOREIGN KEY (child_id) REFERENCES Young_Lives.poverty.child_general_info(child_id);
```

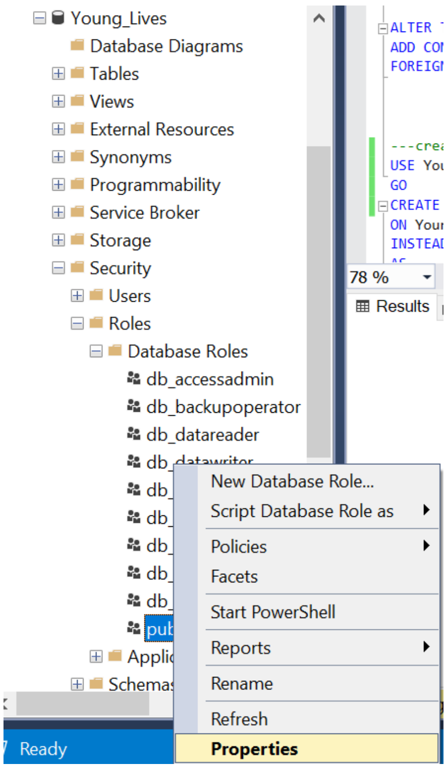
In terms of relational databases, only the first form of normalization is essential (Elmasri and Navathe, 2017). First normal form requires each cell to contain a single value, each column contains the same type of entry and rows need to be unique. The CSV files containing the recorded data for each country already follow this form. Each child is given a unique identifier to identify their responses and the child’s data for each round is recorded in separate rows (Briones, 2018). The use of unique identifiers and round numbers ensures that each row is unique. Additionally, each column only contains a set value which was determined as part of the study’s design. Additional normal forms are optional (Elmasri and Navathe, 2017). Second normal form requires all attributes which are not key columns to be dependent on the key. In this instance the primary key is the child’s unique identifier. All columns within the tables relate to the child’s ID as these are answers which relate to the child’s individual life. Columns cannot be grouped together in other ways which do not relate to the child’s identifier. This also meets the third normal form requirements of all columns can only be determined by the table key.

In multi-user systems, many different users can access the database at the same time. This can create issues with database concurrency if two transactions (units of work) run at the same time (Connolly and Begg, 2015). For example, the lost update problem is caused when two transactions are interleaved, and one transaction reads the data just before another transaction updates the data. This means the transaction will contain old data and so if this transaction also updates the data, any changes made by the previous will be lost. Another issue involves the temporary update (dirty read) problem, where an update made fails and so requires rolling back to the previous data but prior to rolling back this updated data is read by another transaction. The incorrect summary problem involves aggregating transactions obtaining a mixture of updated data and yet-to-be updated data, so is neither an accurate summary of the old data or the updated data. The unrepeatable read problem is caused when a transaction conducts two reads with another transaction updating the value in between, so the transaction reads two different values. Overall, this leads to loss of data integrity and validity. A number of algorithms have been developed to allow transaction concurrency to occur without compromising the data, each with their own strengths and limitations (Elmasri and Navathe, 2017). The method chosen for this database is snapshot isolation, which is a form of optimistic concurrency control. This method involves taking a ‘snapshot’ of the data at the start of the transaction which transactions are conducted on rather than the actual data and once finished any updates can be applied. This method allows reads to occur without needing read ‘locks’, which only allow one read transaction to look at the data at one time, so all read transactions can occur simultaneously (Elmasri and Navathe, 2017). Any transactions which are making any changes to the data will still require locks. This system is useful for this database as the data has already been pre-recorded for the study so should require further editing or updating. It is likely that users will just be using read transactions. In SQL, this concurrency control method is activated within the database using the code shown below (Mukherjee, 2019). Transactions will either be committed if the query is successful or rolled back so changes are not committed to the table.

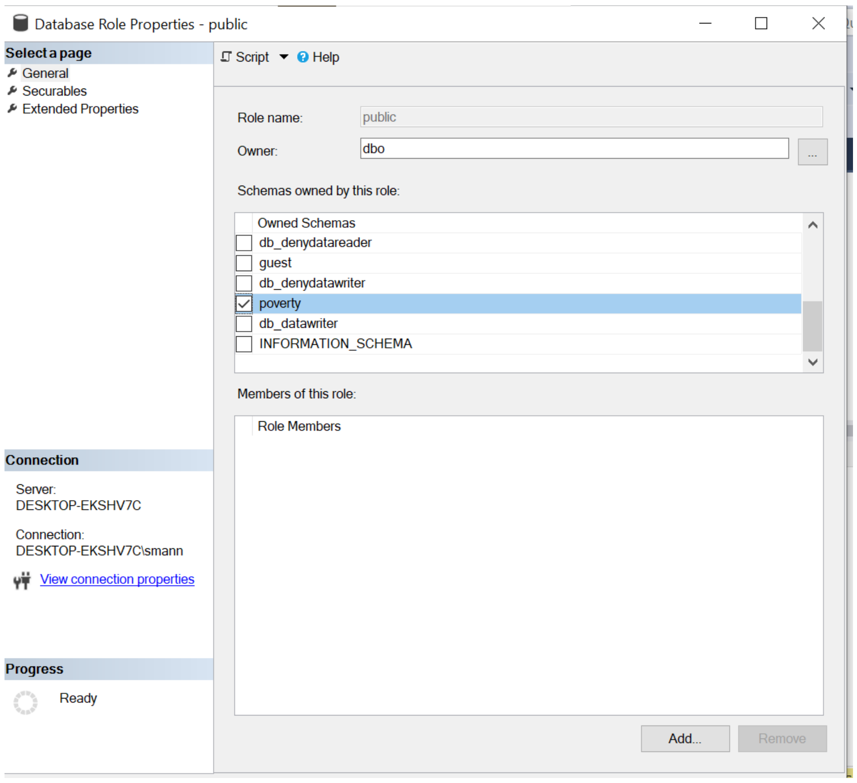


```
--allow for snapshot isolation concurrency method
ALTER DATABASE Young_Lives  SET ALLOW_SNAPSHOT_ISOLATION ON;
ALTER DATABASE Young_Lives SET READ_COMMITTED_SNAPSHOT ON;
```

Many users are expected to be able to use the database to access data, which can put the data at risk of damage (whether intentional or unintentional) or leaking of protected information. Fortunately in the case of this study any private details regarding the children have been anonymised (Briones, 2018). There is still the risk of damage to the data held in the database which would impact the ability to use the data for its intended purpose or for data to be leaked to the outside and used without permission by others. Restricting the database access for individuals in a way which they are can only access what is required to complete their task helps to secure the database by reducing the number of individuals with complete access (Connolly and Begg, 2015). Within the database, users can be created and attached to a particular login. There are two options for users to login to the SQL database, using SQL server login or Windows authentication. It is recommended by Microsoft to use Windows authentication as this is more secure (Mukherjee, 2019). Users can then be given particular privileges which restrict them from accessing all areas of the database and restrict what they are able to do within the database, such as being able to update or change the data in any way. If there are many users, it is probably better to assign users to a role where anyone with the same role has the same privileges. All users are automatically assigned the public role, so this should be the lowest level of access a user is awarded. The public role for this database was only given access to the poverty schema and was not allowed to view/update/alter tables. Views are an excellent method of security, as they only provide individuals access to the attributes required for them to do their job, so any sensitive information in the tables can be withheld (Connolly and Begg, 2015). Therefore, views were used to select data which users would need to investigate certain aspects of child poverty, without giving them access to all the tables.



```
ALTER
ADD CO
FOREIGN

---cre
USE You
GO
CREATE
ON You
INSTEAD
```


Database Role Properties - public

Select a page
- General
- Securables
- Extended Properties

Script ▾ Help

Role name: public

Owner: dbo

Schemas owned by this role:

	Owned Schemas
☐	db_denydatareader
☐	guest
☐	db_denydatawriter
✓	poverty
☐	db_datawriter
☐	INFORMATION_SCHEMA

Members of this role:

Role Members

Add... Remove

Connection

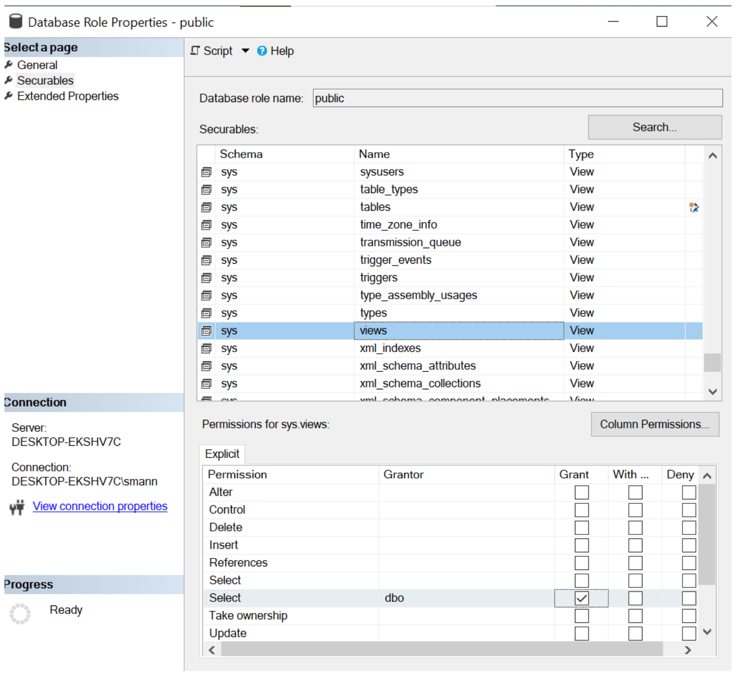
Server:
DESKTOP-EKSHV7C

Connection:
DESKTOP-EKSHV7C\smann

View connection properties

Progress

Ready


Database Role Properties - public

Select a page
- General
- Securables
- Extended Properties

Script ▾ Help

Database role name: public

Securables: Search...

Schema	Name	Type
sys	sysusers	View
sys	table_types	View
sys	tables	View
sys	time_zone_info	View
sys	transmission_queue	View
sys	trigger_events	View
sys	triggers	View
sys	type_assembly_usages	View
sys	types	View
sys	views	View
sys	xml_indexes	View
sys	xml_schema_attributes	View
sys	xml_schema_collections	View

Permissions for sys.views: Column Permissions...

Explicit

Permission	Grantor	Grant	With ...	Deny
Alter		☐	☐	☐
Control		☐	☐	☐
Delete		☐	☐	☐
Insert		☐	☐	☐
References		☐	☐	☐
Select		☐	☐	☐
Select	dbo	✓	☐	☐
Take ownership		☐	☐	☐
Update		☐	☐	☐

Connection

Server:
DESKTOP-EKSHV7C

Connection:
DESKTOP-EKSHV7C\smann

View connection properties

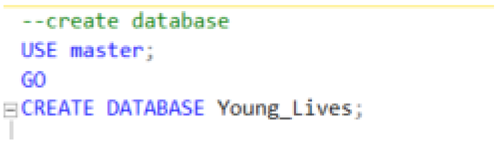
Progress

Ready

As multiple users within a database may have privileges which allow them to make edits to the database, including inserting data, creating new functions or stored procedures, etc., it is important to make comments within SQL code to explain the function of sections of code and to explain any changes made to sections of code (Mukherjee, 2019). Additionally, it can be useful to block out sections of code which do not need to be run but may be required in a later instance. Although not applied here, it is good convention to mark any changes to code with comments to explain the changes, the date of change and who created the change, particularly if there are multiple users with the privileges to make database changes.

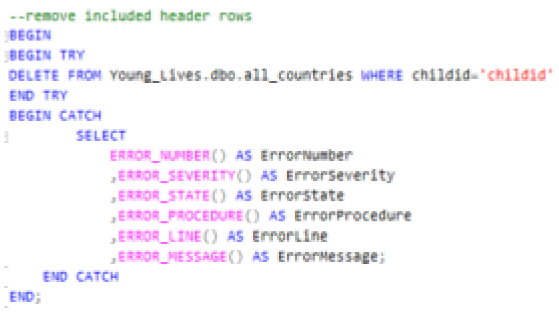
5. T-SQL statements

A new database was created called ‘Young_Lives’ which would contain the data from the study. The CSV files for each country were imported into the ‘Young_Lives’ database using SSMS’s Import and Export Wizard, which generates a table automatically, with the first row used as the column attributes and the delimiter set used to separate data into appropriate columns. Following this the different country tables were appended together using ‘UNION ALL’ as they contained the same columns, creating a table which contained all country data. The union of tables caused column titles to be made into rows within the table, so any columns which contained the string ‘child_id’ were removed.



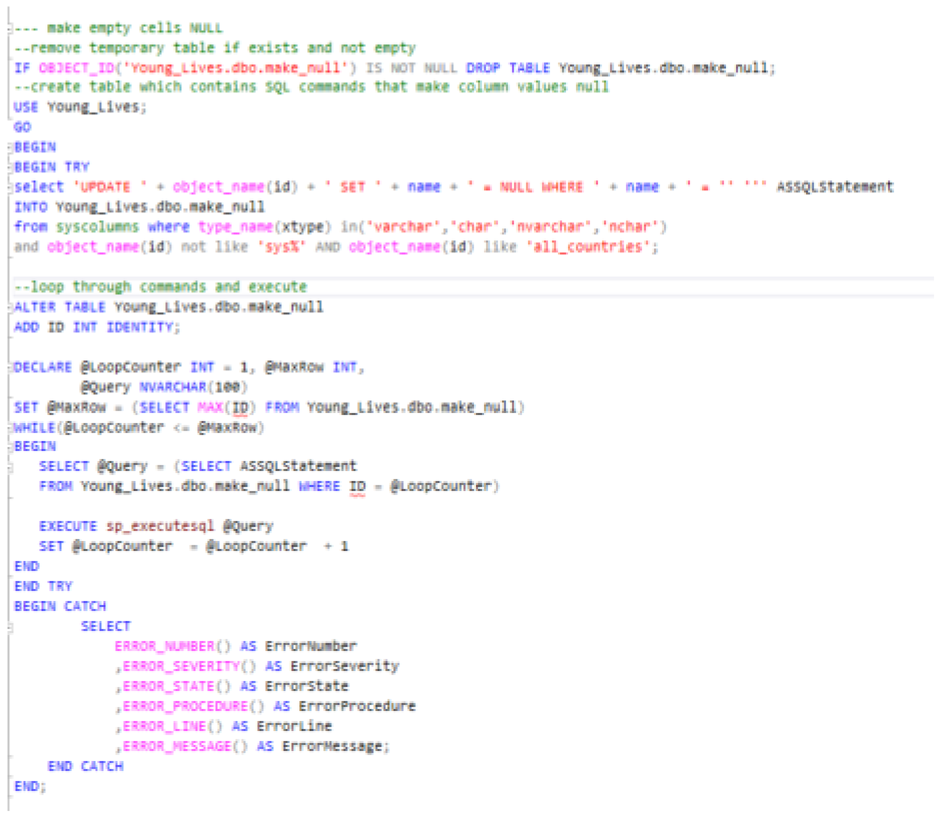
```
--create database
USE master;
GO
CREATE DATABASE Young_Lives;
```

[illegible]



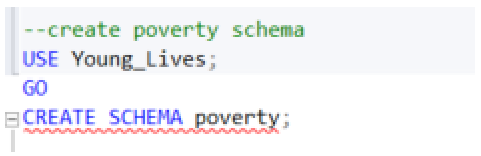
```sql
--remove included header rows
BEGIN
BEGIN TRY
DELETE FROM Young_Lives.dbo.all_countries WHERE childid='childid'
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

The data provided was all in numeric format and so needed transforming into meaningful data to allow interpretation when building reports. Prior to this, all columns which did not contain any values needed to be transformed to NULL values. This is because when casting columns with numeric data types, it is not possible when blank values are considered strings and so raises an error.

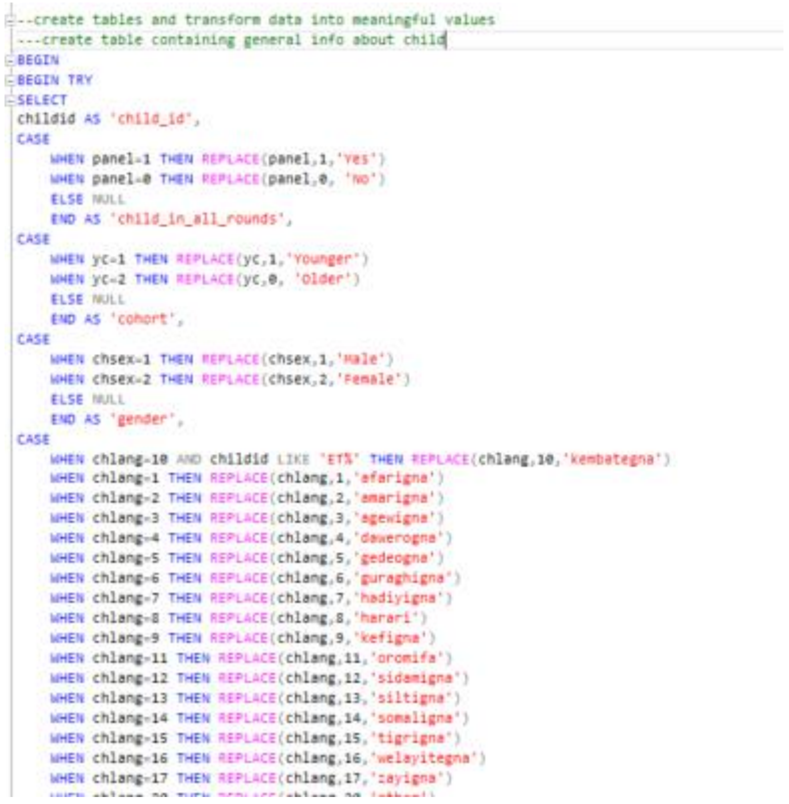


```sql
--- make empty cells NULL
--remove temporary table if exists and not empty
IF OBJECT_ID('Young_Lives.dbo.make_null') IS NOT NULL DROP TABLE Young_Lives.dbo.make_null;
--create table which contains SQL commands that make column values null
USE Young_Lives;
GO
BEGIN
BEGIN TRY
select 'UPDATE ' + object_name(id) + ' SET ' + name + ' = NULL WHERE ' + name + ' = '' ''' ASSQLStatement
INTO Young_Lives.dbo.make_null
from syscolumns where type_name(xtype) in('varchar','char','nvarchar','nchar')
and object_name(id) not like 'sys%' AND object_name(id) like 'all_countries';

--loop through commands and execute
ALTER TABLE Young_Lives.dbo.make_null
ADD ID INT IDENTITY;

DECLARE @LoopCounter INT = 1, @MaxRow INT,
        @Query NVARCHAR(100)
SET @MaxRow = (SELECT MAX(ID) FROM Young_Lives.dbo.make_null)
WHILE(@LoopCounter <= @MaxRow)
BEGIN
   SELECT @Query = (SELECT ASSQLStatement
   FROM Young_Lives.dbo.make_null WHERE ID = @LoopCounter)

   EXECUTE sp_executesql @Query
   SET @LoopCounter  = @LoopCounter  + 1
END
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

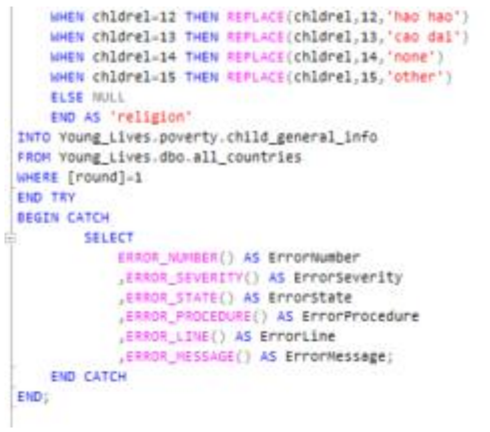
Schema creation within a database can add an extra layer of security by ensuring certain users can only access particular schemas that are relevant to their work (Mukherjee, 2019). Therefore, when creating the tables with the purpose of measuring child poverty, these were grouped into the newly created schema ‘poverty’. The data from the table containing all countries was split into seven tables, as seen in the created relational schema diagram. Columns for each table were selected and transformed into meaningful data using the commands ‘CASE WHEN’ and then inserted into new tables. One table contained information about the child which was consistent across rounds and so only round one values were inserted and ‘child_id’ was set as the primary key. This table was called ‘child_general_info’. Other tables all contained the columns ‘child_id’ and ‘round’ along with data which could be grouped based on similarity. To ensure data essential data was not lost from the main table (‘child_general_info’), a trigger was created which stops any deletion from occurring on this table and prints an error message. This ensures data is not lost in dependent tables and views, as well as protects data accuracy and integrity by preventing crime reports from being removed.



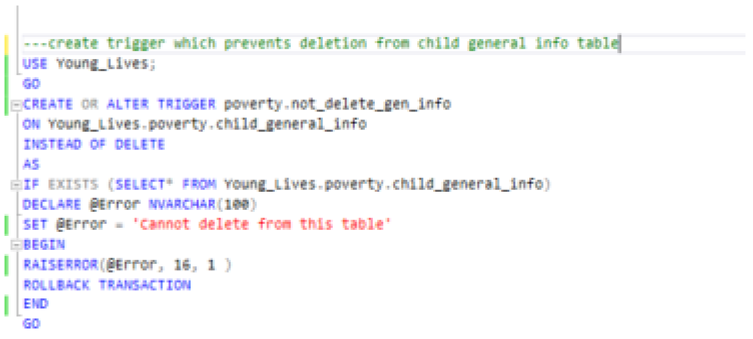
```sql
--create poverty schema
USE Young_Lives;
GO
CREATE SCHEMA poverty;
```



```sql
--create tables and transform data into meaningful values
---create table containing general info about child
BEGIN
BEGIN TRY
SELECT
childid AS 'child_id',
CASE
    WHEN panel=1 THEN REPLACE(panel,1,'Yes')
    WHEN panel=0 THEN REPLACE(panel,0, 'No')
    ELSE NULL
    END AS 'child_in_all_rounds',
CASE
    WHEN yc=1 THEN REPLACE(yc,1,'Younger')
    WHEN yc=2 THEN REPLACE(yc,0, 'Older')
    ELSE NULL
    END AS 'cohort',
CASE
    WHEN chsex=1 THEN REPLACE(chsex,1,'Male')
    WHEN chsex=2 THEN REPLACE(chsex,2,'Female')
    ELSE NULL
    END AS 'gender',
CASE
    WHEN chlang=10 AND childid LIKE 'ET%' THEN REPLACE(chlang,10,'kembategna')
    WHEN chlang=1 THEN REPLACE(chlang,1,'afarigna')
    WHEN chlang=2 THEN REPLACE(chlang,2,'amarigna')
    WHEN chlang=3 THEN REPLACE(chlang,3,'agewigna')
    WHEN chlang=4 THEN REPLACE(chlang,4,'dawerogna')
    WHEN chlang=5 THEN REPLACE(chlang,5,'gedeogna')
    WHEN chlang=6 THEN REPLACE(chlang,6,'guraghigna')
    WHEN chlang=7 THEN REPLACE(chlang,7,'hadiyigna')
    WHEN chlang=8 THEN REPLACE(chlang,8,'harari')
    WHEN chlang=9 THEN REPLACE(chlang,9,'kefigna')
    WHEN chlang=11 THEN REPLACE(chlang,11,'oromifa')
    WHEN chlang=12 THEN REPLACE(chlang,12,'sidamigna')
    WHEN chlang=13 THEN REPLACE(chlang,13,'siltigna')
    WHEN chlang=14 THEN REPLACE(chlang,14,'somaligna')
    WHEN chlang=15 THEN REPLACE(chlang,15,'tigrigna')
    WHEN chlang=16 THEN REPLACE(chlang,16,'welayitegna')
    WHEN chlang=17 THEN REPLACE(chlang,17,'zayigna')
    WHEN chlang=20 THEN REPLACE(chlang,20,'other')
```

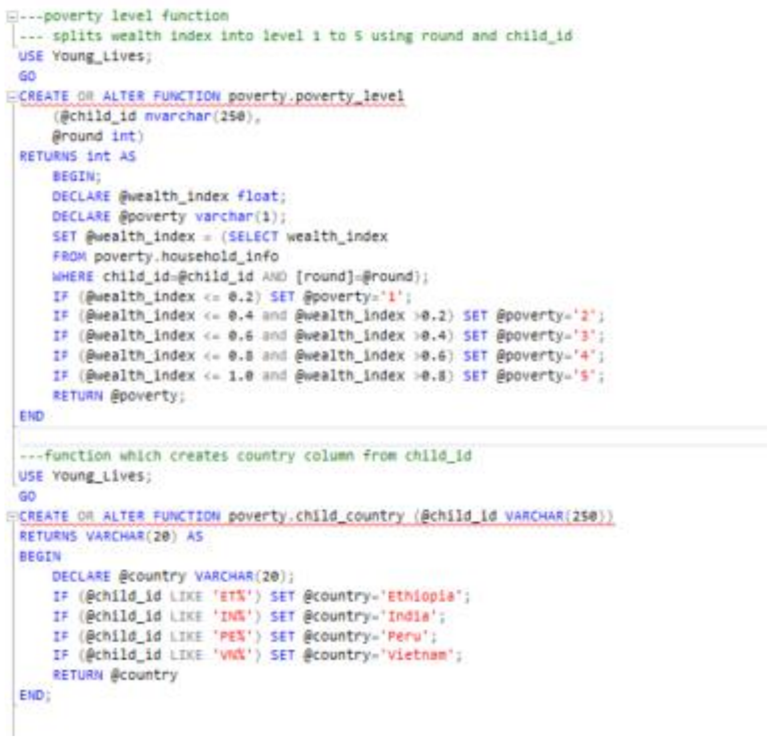


```sql
    WHEN chldrel=12 THEN REPLACE(chldrel,12,'hao hao')
    WHEN chldrel=13 THEN REPLACE(chldrel,13,'cao dai')
    WHEN chldrel=14 THEN REPLACE(chldrel,14,'none')
    WHEN chldrel=15 THEN REPLACE(chldrel,15,'other')
    ELSE NULL
    END AS 'religion'
INTO Young_Lives.poverty.child_general_info
FROM Young_Lives.dbo.all_countries
WHERE [round]=1
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```



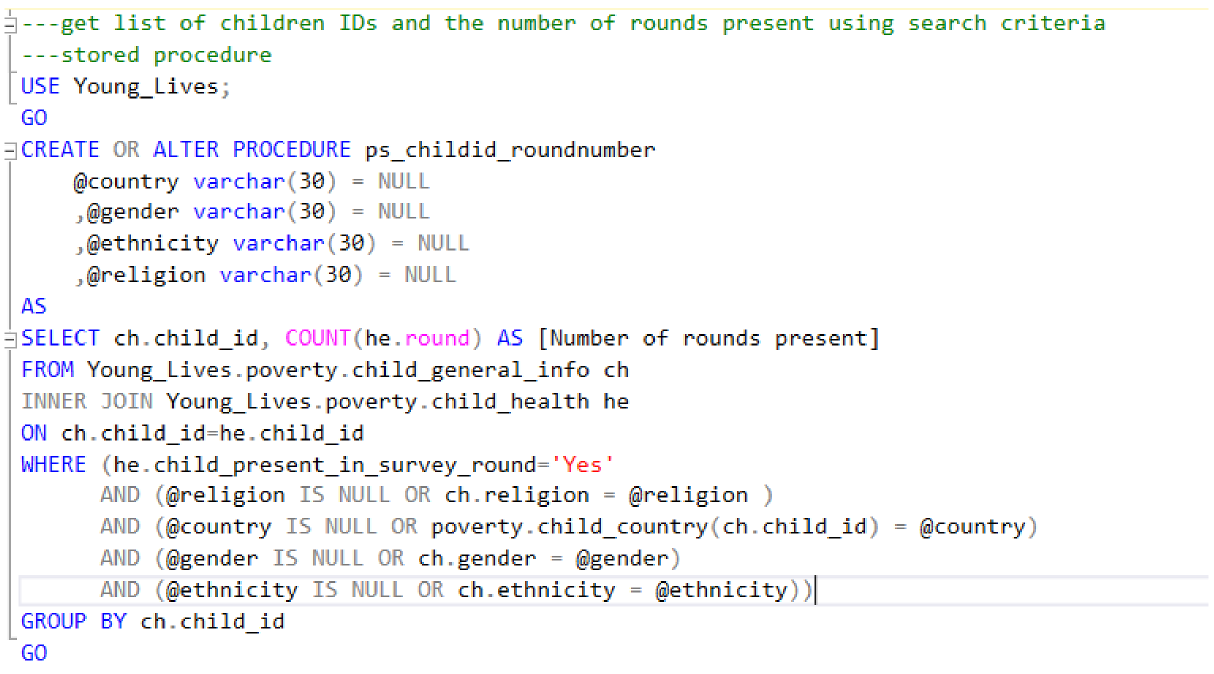
```sql
---create trigger which prevents deletion from child general info table
USE Young_Lives;
GO
CREATE OR ALTER TRIGGER poverty.not_delete_gen_info
ON Young_Lives.poverty.child_general_info
INSTEAD OF DELETE
AS
IF EXISTS (SELECT* FROM Young_Lives.poverty.child_general_info)
DECLARE @Error NVARCHAR(100)
SET @Error = 'Cannot delete from this table'
BEGIN
RAISERROR(@Error, 16, 1 )
ROLLBACK TRANSACTION
END
GO
```

Example views were created to summarise data which would allow investigation and analysis into differences relating to child poverty. In the study, each round the child's family were given a score between 0 and 1 which indicated their level of wealth, with 0 being the lowest. A function was created which, when provided the child's ID and round, grouped children into five wealth groups according to their wealth index, with 1 being the poorest and 5 being the wealthiest. Additionally, as the country the child was from was indicated by the first two letters of their unique ID, another function was created which when provided with the child's ID would return the child's country.



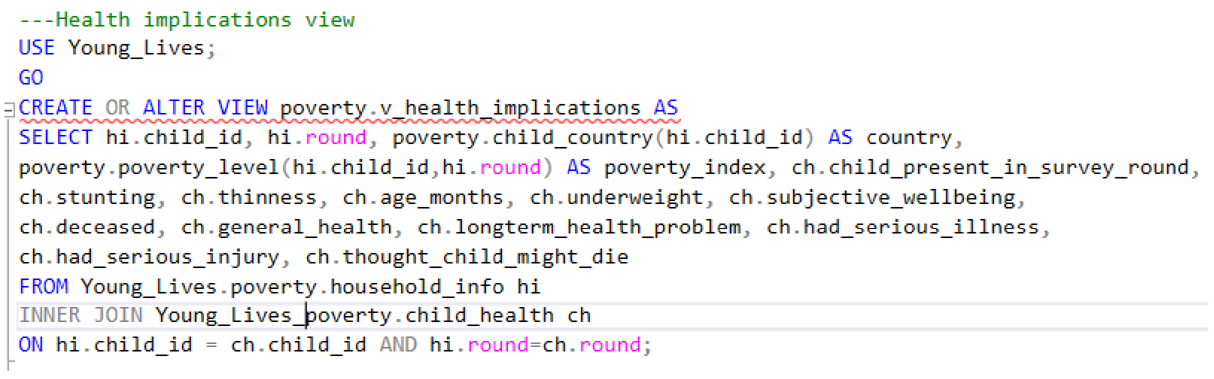
```sql
---poverty level function
--- splits wealth index into level 1 to 5 using round and child_id
USE Young_Lives;
GO
CREATE OR ALTER FUNCTION poverty.poverty_level
    (@child_id nvarchar(250),
    @round int)
RETURNS int AS
    BEGIN;
    DECLARE @wealth_index float;
    DECLARE @poverty varchar(1);
    SET @wealth_index = (SELECT wealth_index
    FROM poverty.household_info
    WHERE child_id=@child_id AND [round]=@round);
    IF (@wealth_index <= 0.2) SET @poverty='1';
    IF (@wealth_index <= 0.4 and @wealth_index >0.2) SET @poverty='2';
    IF (@wealth_index <= 0.6 and @wealth_index >0.4) SET @poverty='3';
    IF (@wealth_index <= 0.8 and @wealth_index >0.6) SET @poverty='4';
    IF (@wealth_index <= 1.0 and @wealth_index >0.8) SET @poverty='5';
    RETURN @poverty;
END

---function which creates country column from child_id
USE Young_Lives;
GO
CREATE OR ALTER FUNCTION poverty.child_country (@child_id VARCHAR(250))
RETURNS VARCHAR(20) AS
BEGIN
    DECLARE @country VARCHAR(20);
    IF (@child_id LIKE 'ET%') SET @country='Ethiopia';
    IF (@child_id LIKE 'IN%') SET @country='India';
    IF (@child_id LIKE 'PE%') SET @country='Peru';
    IF (@child_id LIKE 'VN%') SET @country='Vietnam';
    RETURN @country
END;
```

As the study was conducted over multiple rounds, some children did not participate in all rounds, which is only indicated by the column 'child_present_in_round'. Researchers are likely to want to optimise their sample if looking at the longitudinal effects of child poverty and preferentially select those who are present in all rounds or at least most rounds. The stored procedure created called 'ps_childid_roundnumber' allows users to search for particular characteristics they may wish the sample to contain, such as specific country, particular gender, particular ethnicity, present for a specific number of rounds, etc. The results produced are the child IDs and number of rounds which correspond to these criteria allowing users to quickly generate desired samples they may wish to use for analysis.

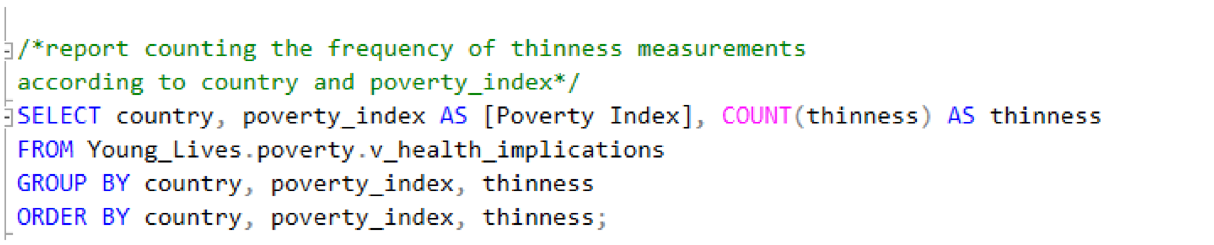


```sql
---get list of children IDs and the number of rounds present using search criteria
---stored procedure
USE Young_Lives;
GO
CREATE OR ALTER PROCEDURE ps_childid_roundnumber
    @country varchar(30) = NULL
    ,@gender varchar(30) = NULL
    ,@ethnicity varchar(30) = NULL
    ,@religion varchar(30) = NULL
AS
SELECT ch.child_id, COUNT(he.round) AS [Number of rounds present]
FROM Young_Lives.poverty.child_general_info ch
INNER JOIN Young_Lives.poverty.child_health he
ON ch.child_id=he.child_id
WHERE (he.child_present_in_survey_round='Yes'
      AND (@religion IS NULL OR ch.religion = @religion )
      AND (@country IS NULL OR poverty.child_country(ch.child_id) = @country)
      AND (@gender IS NULL OR ch.gender = @gender)
      AND (@ethnicity IS NULL OR ch.ethnicity = @ethnicity))
GROUP BY ch.child_id
GO
```

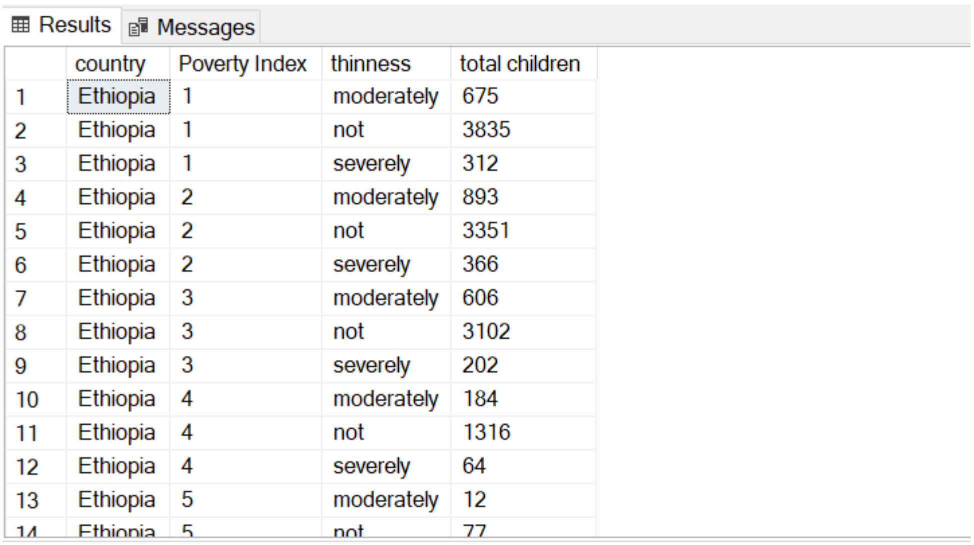
In previous research relating to the data gathered in the Young Lives study, poorer children have been found to experience more illness, which affects their weight, height and well-being (Pells, 2011b). Often poorer families are unable to access adequate healthcare. Therefore, health-related data and the child's family wealth index were stored in a view called 'v_health_implications. Example reports were provided which allow summarisation of how poverty influences a child's health, by counting the number of children who experienced thinness, stunted growth and were underweight for each wealth group ('poverty_index') and per country, as different countries will likely have different prevalent health issues and access to healthcare systems. Another report produced the proportion of rounds a child had had a long-term health condition, had a serious injury or illness, or caregivers had thought the child might die along with the child's average poverty index. As children were present for a different number of rounds, the count of rounds the child was present for was used to work out the proportion.



```sql
---Health implications view
USE Young_Lives;
GO
CREATE OR ALTER VIEW poverty.v_health_implications AS
SELECT hi.child_id, hi.round, poverty.child_country(hi.child_id) AS country,
poverty.poverty_level(hi.child_id,hi.round) AS poverty_index, ch.child_present_in_survey_round,
ch.stunting, ch.thinness, ch.age_months, ch.underweight, ch.subjective_wellbeing,
ch.deceased, ch.general_health, ch.longterm_health_problem, ch.had_serious_illness,
ch.had_serious_injury, ch.thought_child_might_die
FROM Young_Lives.poverty.household_info hi
INNER JOIN Young_Lives.poverty.child_health ch
ON hi.child_id = ch.child_id AND hi.round=ch.round;
```

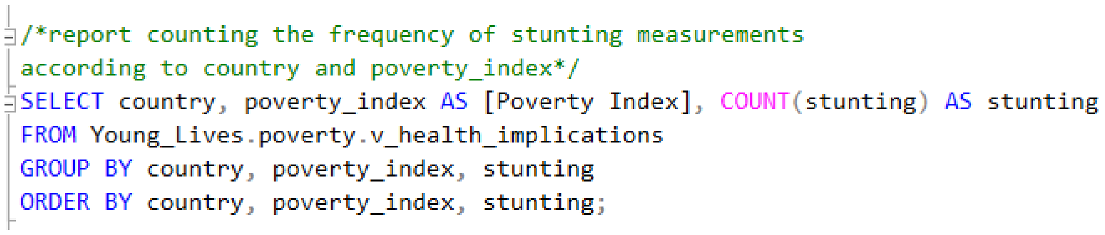


```sql
/*report counting the frequency of thinness measurements
according to country and poverty_index*/
SELECT country, poverty_index AS [Poverty Index], COUNT(thinness) AS thinness
FROM Young_Lives.poverty.v_health_implications
GROUP BY country, poverty_index, thinness
ORDER BY country, poverty_index, thinness;
```

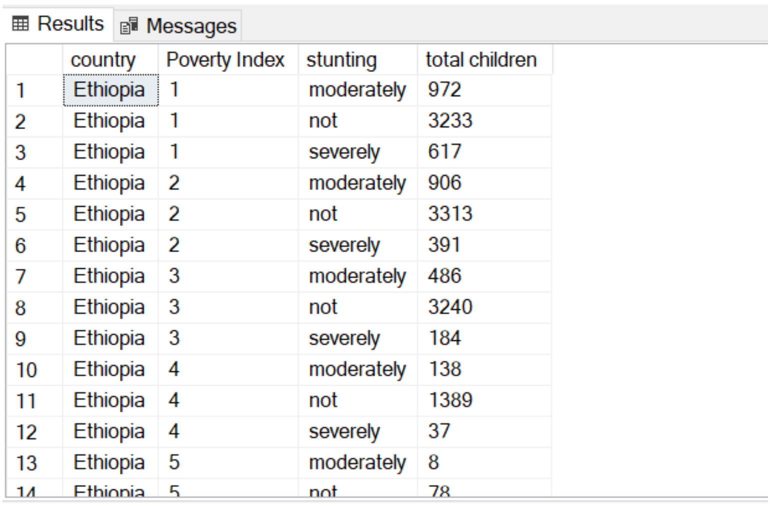


Results | Messages

	country	Poverty Index	thinness	total children
1	Ethiopia	1	moderately	675
2	Ethiopia	1	not	3835
3	Ethiopia	1	severely	312
4	Ethiopia	2	moderately	893
5	Ethiopia	2	not	3351
6	Ethiopia	2	severely	366
7	Ethiopia	3	moderately	606
8	Ethiopia	3	not	3102
9	Ethiopia	3	severely	202
10	Ethiopia	4	moderately	184
11	Ethiopia	4	not	1316
12	Ethiopia	4	severely	64
13	Ethiopia	5	moderately	12
14	Ethiopia	5	not	77

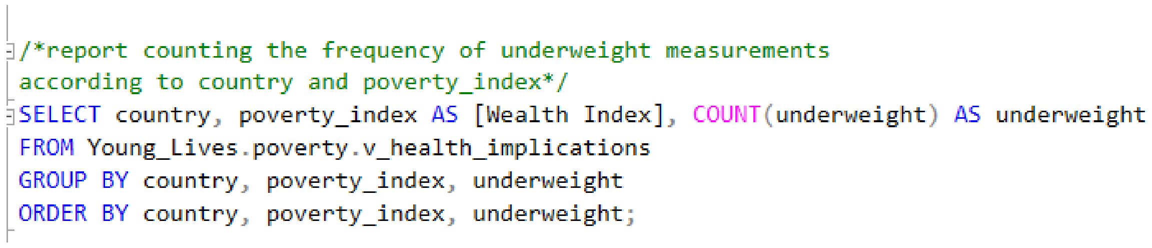


```
/*report counting the frequency of stunting measurements
according to country and poverty_index*/
SELECT country, poverty_index AS [Poverty Index], COUNT(stunting) AS stunting
FROM Young_Lives.poverty.v_health_implications
GROUP BY country, poverty_index, stunting
ORDER BY country, poverty_index, stunting;
```

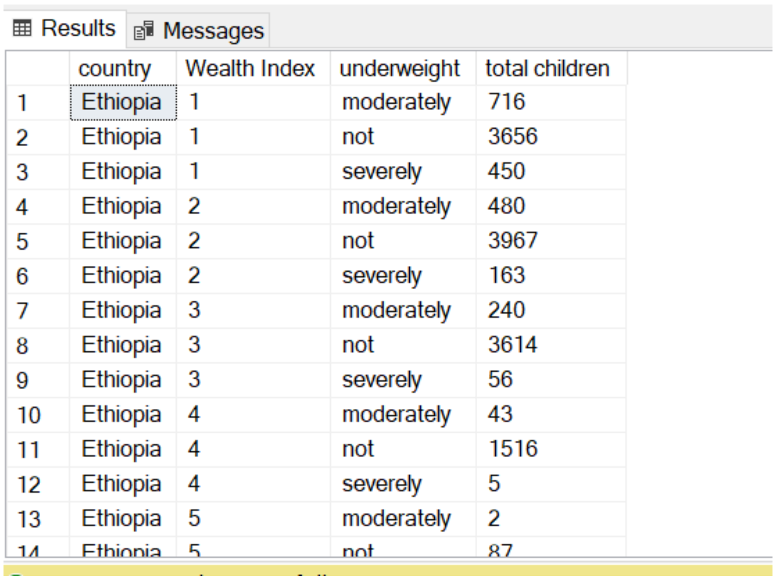


Results | Messages

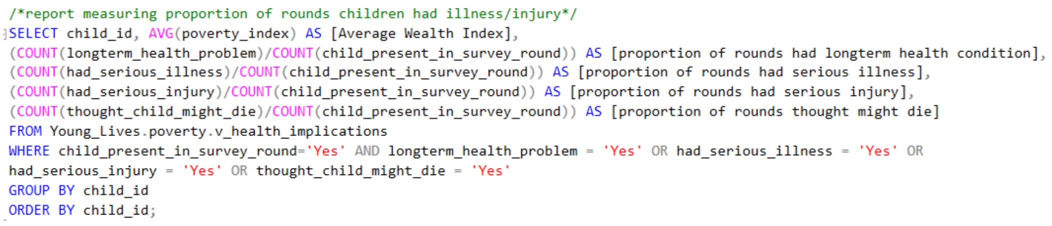
	country	Poverty Index	stunting	total children
1	Ethiopia	1	moderately	972
2	Ethiopia	1	not	3233
3	Ethiopia	1	severely	617
4	Ethiopia	2	moderately	906
5	Ethiopia	2	not	3313
6	Ethiopia	2	severely	391
7	Ethiopia	3	moderately	486
8	Ethiopia	3	not	3240
9	Ethiopia	3	severely	184
10	Ethiopia	4	moderately	138
11	Ethiopia	4	not	1389
12	Ethiopia	4	severely	37
13	Ethiopia	5	moderately	8
14	Ethiopia	5	not	78

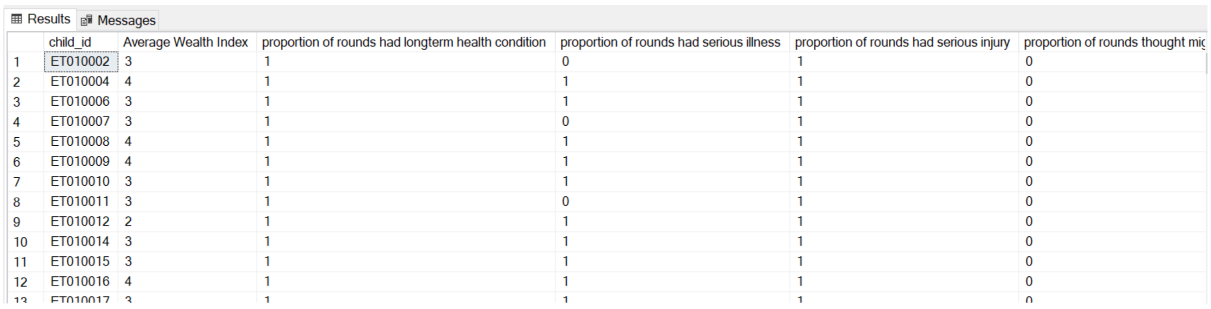


```
/*report counting the frequency of underweight measurements
according to country and poverty_index*/
SELECT country, poverty_index AS [Wealth Index], COUNT(underweight) AS underweight
FROM Young_Lives.poverty.v_health_implications
GROUP BY country, poverty_index, underweight
ORDER BY country, poverty_index, underweight;
```


Results | Messages

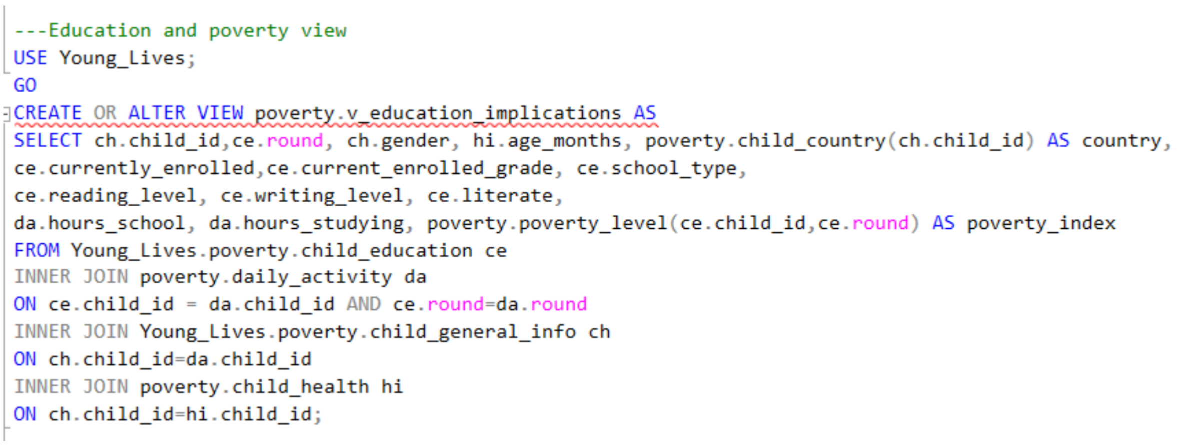
	country	Wealth Index	underweight	total children
1	Ethiopia	1	moderately	716
2	Ethiopia	1	not	3656
3	Ethiopia	1	severely	450
4	Ethiopia	2	moderately	480
5	Ethiopia	2	not	3967
6	Ethiopia	2	severely	163
7	Ethiopia	3	moderately	240
8	Ethiopia	3	not	3614
9	Ethiopia	3	severely	56
10	Ethiopia	4	moderately	43
11	Ethiopia	4	not	1516
12	Ethiopia	4	severely	5
13	Ethiopia	5	moderately	2
14	Ethiopia	5	not	87


```
/*report measuring proportion of rounds children had illness/injury*/
SELECT child_id, AVG(poverty_index) AS [Average Wealth Index],
(COUNT(longterm_health_problem)/COUNT(child_present_in_survey_round)) AS [proportion of rounds had longterm health condition],
(COUNT(had_serious_illness)/COUNT(child_present_in_survey_round)) AS [proportion of rounds had serious illness],
(COUNT(had_serious_injury)/COUNT(child_present_in_survey_round)) AS [proportion of rounds had serious injury],
(COUNT(thought_child_might_die)/COUNT(child_present_in_survey_round)) AS [proportion of rounds thought might die]
FROM Young_Lives.poverty.v_health_implications
WHERE child_present_in_survey_round='Yes' AND longterm_health_problem = 'Yes' OR had_serious_illness = 'Yes' OR
had_serious_injury = 'Yes' OR thought_child_might_die = 'Yes'
GROUP BY child_id
ORDER BY child_id;
```

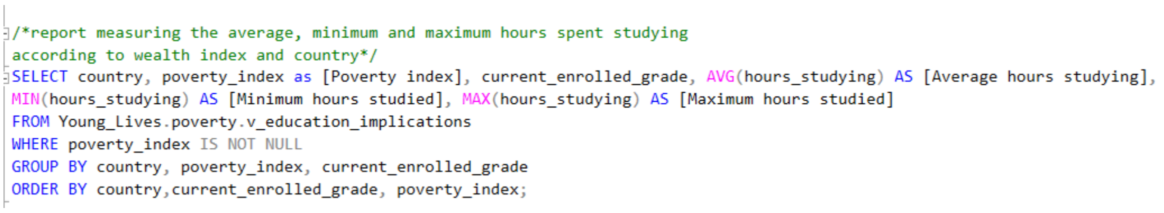

Results | Messages

	child_id	Average Wealth Index	proportion of rounds had longterm health condition	proportion of rounds had serious illness	proportion of rounds had serious injury	proportion of rounds thought mig
1	ET010002	3	1	0	1	0
2	ET010004	4	1	1	1	0
3	ET010006	3	1	1	1	0
4	ET010007	3	1	0	1	0
5	ET010008	4	1	1	1	0
6	ET010009	4	1	1	1	0
7	ET010010	3	1	1	1	0
8	ET010011	3	1	0	1	0
9	ET010012	2	1	1	1	0
10	ET010014	3	1	1	1	0
11	ET010015	3	1	1	1	0
12	ET010016	4	1	1	1	0
13	ET010017	3	1	1	1	0

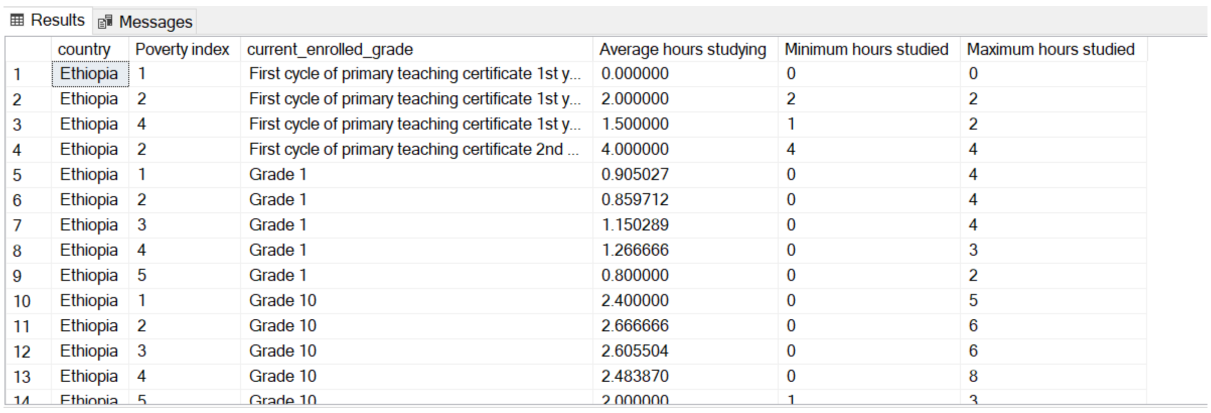
As poverty is known to influence academic achievement in children, a view containing education measurements with poverty index. Poorer children likely to spend less time studying (Pells, 2011a). The average, minimum and maximum hours spent studying for each wealth group according to country was reported. As the grade of the student could affect the amount of studying committed this was also used to group data. Additionally, poorer children are more likely to drop out of education earlier (Pells, 2011a, Morrow and Pells, 2012). The percentage of children who dropped out of education prior to the age of 16 at some point during the study for each poverty index and country was reported. As some children were older and so would obviously no longer be enrolled in school, the age of unenrolment had to be less than 16.



```
---Education and poverty view
USE Young_Lives;
GO
CREATE OR ALTER VIEW poverty.v_education_implications AS
SELECT ch.child_id,ce.round, ch.gender, hi.age_months, poverty.child_country(ch.child_id) AS country,
ce.currently_enrolled,ce.current_enrolled_grade, ce.school_type,
ce.reading_level, ce.writing_level, ce.literate,
da.hours_school, da.hours_studying, poverty.poverty_level(ce.child_id,ce.round) AS poverty_index
FROM Young_Lives.poverty.child_education ce
INNER JOIN poverty.daily_activity da
ON ce.child_id = da.child_id AND ce.round=da.round
INNER JOIN Young_Lives.poverty.child_general_info ch
ON ch.child_id=da.child_id
INNER JOIN poverty.child_health hi
ON ch.child_id=hi.child_id;
```

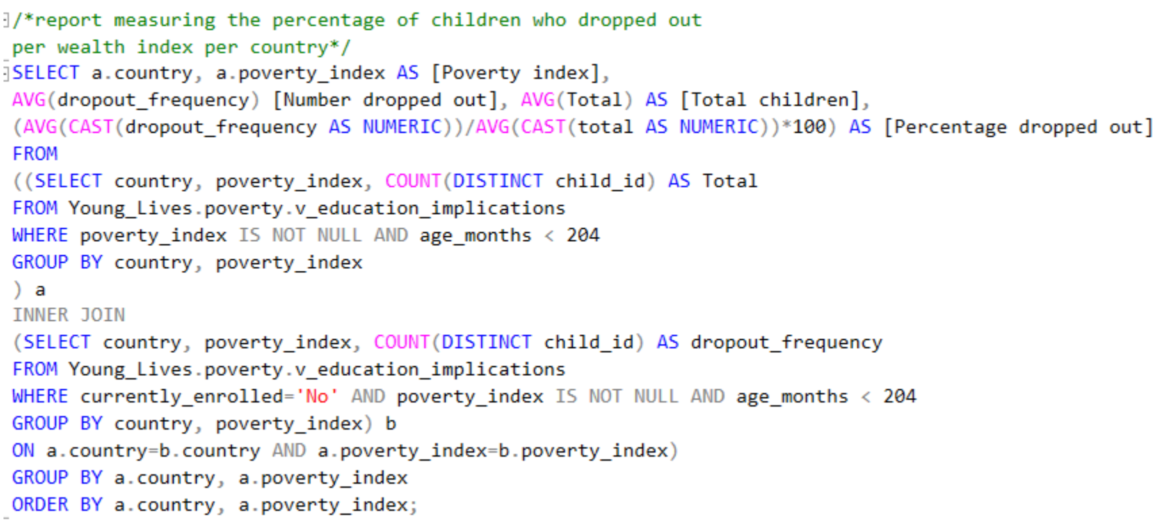


```
/*report measuring the average, minimum and maximum hours spent studying
according to wealth index and country*/
SELECT country, poverty_index as [Poverty index], current_enrolled_grade, AVG(hours_studying) AS [Average hours studying],
MIN(hours_studying) AS [Minimum hours studied], MAX(hours_studying) AS [Maximum hours studied]
FROM Young_Lives.poverty.v_education_implications
WHERE poverty_index IS NOT NULL
GROUP BY country, poverty_index, current_enrolled_grade
ORDER BY country,current_enrolled_grade, poverty_index;
```

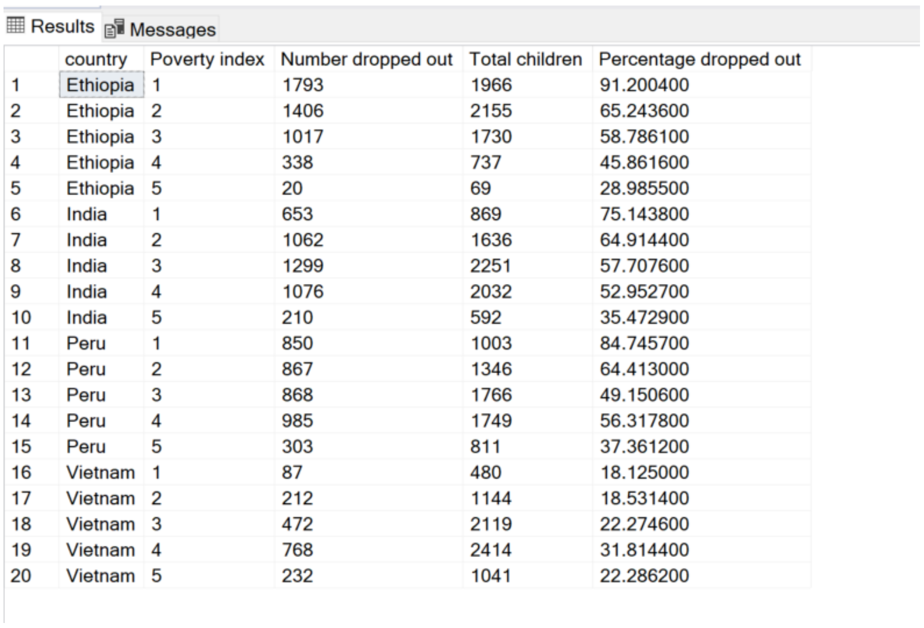


Results | Messages

	country	Poverty index	current_enrolled_grade	Average hours studying	Minimum hours studied	Maximum hours studied
1	Ethiopia	1	First cycle of primary teaching certificate 1st y...	0.000000	0	0
2	Ethiopia	2	First cycle of primary teaching certificate 1st y...	2.000000	2	2
3	Ethiopia	4	First cycle of primary teaching certificate 1st y...	1.500000	1	2
4	Ethiopia	2	First cycle of primary teaching certificate 2nd ...	4.000000	4	4
5	Ethiopia	1	Grade 1	0.905027	0	4
6	Ethiopia	2	Grade 1	0.859712	0	4
7	Ethiopia	3	Grade 1	1.150289	0	4
8	Ethiopia	4	Grade 1	1.266666	0	3
9	Ethiopia	5	Grade 1	0.800000	0	2
10	Ethiopia	1	Grade 10	2.400000	0	5
11	Ethiopia	2	Grade 10	2.666666	0	6
12	Ethiopia	3	Grade 10	2.605504	0	6
13	Ethiopia	4	Grade 10	2.483870	0	8
14	Ethiopia	5	Grade 10	2.000000	1	3



```
/*report measuring the percentage of children who dropped out
per wealth index per country*/
SELECT a.country, a.poverty_index AS [Poverty index],
AVG(dropout_frequency) [Number dropped out], AVG(Total) AS [Total children],
(AVG(CAST(dropout_frequency AS NUMERIC))/AVG(CAST(total AS NUMERIC))*100) AS [Percentage dropped out]
FROM
((SELECT country, poverty_index, COUNT(DISTINCT child_id) AS Total
FROM Young_Lives.poverty.v_education_implications
WHERE poverty_index IS NOT NULL AND age_months < 204
GROUP BY country, poverty_index
) a
INNER JOIN
(SELECT country, poverty_index, COUNT(DISTINCT child_id) AS dropout_frequency
FROM Young_Lives.poverty.v_education_implications
WHERE currently_enrolled='No' AND poverty_index IS NOT NULL AND age_months < 204
GROUP BY country, poverty_index) b
ON a.country=b.country AND a.poverty_index=b.poverty_index)
GROUP BY a.country, a.poverty_index
ORDER BY a.country, a.poverty_index;
```


Results Messages

	country	Poverty index	Number dropped out	Total children	Percentage dropped out
1	Ethiopia	1	1793	1966	91.200400
2	Ethiopia	2	1406	2155	65.243600
3	Ethiopia	3	1017	1730	58.786100
4	Ethiopia	4	338	737	45.861600
5	Ethiopia	5	20	69	28.985500
6	India	1	653	869	75.143800
7	India	2	1062	1636	64.914400
8	India	3	1299	2251	57.707600
9	India	4	1076	2032	52.952700
10	India	5	210	592	35.472900
11	Peru	1	850	1003	84.745700
12	Peru	2	867	1346	64.413000
13	Peru	3	868	1766	49.150600
14	Peru	4	985	1749	56.317800
15	Peru	5	303	811	37.361200
16	Vietnam	1	87	480	18.125000
17	Vietnam	2	212	1144	18.531400
18	Vietnam	3	472	2119	22.274600
19	Vietnam	4	768	2414	31.814400
20	Vietnam	5	232	1041	22.286200

6. Database security

Besides the methods of security which have been applied in the database already, there are other recommended steps which can be taken to improve the security of the database. It is possible to encrypt data to protect it from any threats or attempts to access, particularly if this data is deemed sensitive (Connolly and Begg, 2015). SQL Server offers multiple functions which can be used to encrypt data, such as 'ENCRYPTBYKEY' and 'ENCRYPTBYPASSPHRASE' (Mukherjee, 2019). Besides tools within the database, it is also recommended that external security measures are also taken (Mukherjee, 2019, Elmasri and Navathe, 2017). For example, the hardware used to run SQL server must be fault tolerant. Should one of the components fail the server can continue to run without failure. A potential option is using Redundant Array of Independent Disks (RAID) which allows data to be segmented and distributed across multiple disks (Connolly and Begg, 2015). Additional external security can involve ensuring the operating system used is up-to-date with the latest upgrades to security and that there is an operational firewall in place (Connolly and Begg, 2015). Finally, it is recommended that the hardware has physical security, such as locking the server hardware away and restricting access (Connolly and Begg, 2015).

7. Database backup

Backing up and recovering a database is another form of security. Should a failure occur where the database can no longer be used, a recent back up version can be restored (Connolly and Begg, 2015).

8. Conclusion

Using a database to store the data from the Young Lives study can allow maintenance of data integrity, accuracy, and concurrency. It can also allow multiple users to use the data for future research through allowing users to create their own required combinations of attributes and calculated fields, without impacting the stored data. In the future, Young Lives could store further study data, including extensions of the study used in this task, as well as other related studies conducted as part of the project.

Task 2: School Survey Vietnam 2016-17

Databases offer the opportunity to ensure an organisation's data is maintained through protecting data integrity, data concurrency and security. The storage of data in one central location can allow multiple users to access required data to meet organisation objectives and goals efficiently and quickly. This is especially important with the amount of data generated in the present age. This task aims to import data from the Young Lives school survey into a database, creating logically related tables, with the aim of enhancing researchers' ability to investigate education inequalities in Vietnam.

1. Introduction

For a long time, data was stored independently in separate files. This approach has a lot of drawbacks which can make business use inefficient (Connolly and Begg, 2015). Linking related data kept in separate files required creating new files to store combined data. Data could often be duplicated across departments and teams. Any updating of data could require updating data across multiple files, which will likely create data inconsistencies and future errors. In the end, any novel user requirements of the data would have to be custom-written and produce by the IT team, increasing the amount of time business tasks could take. With the increasing amounts of data produced, this system would only become more difficult (Connolly and Begg, 2015).

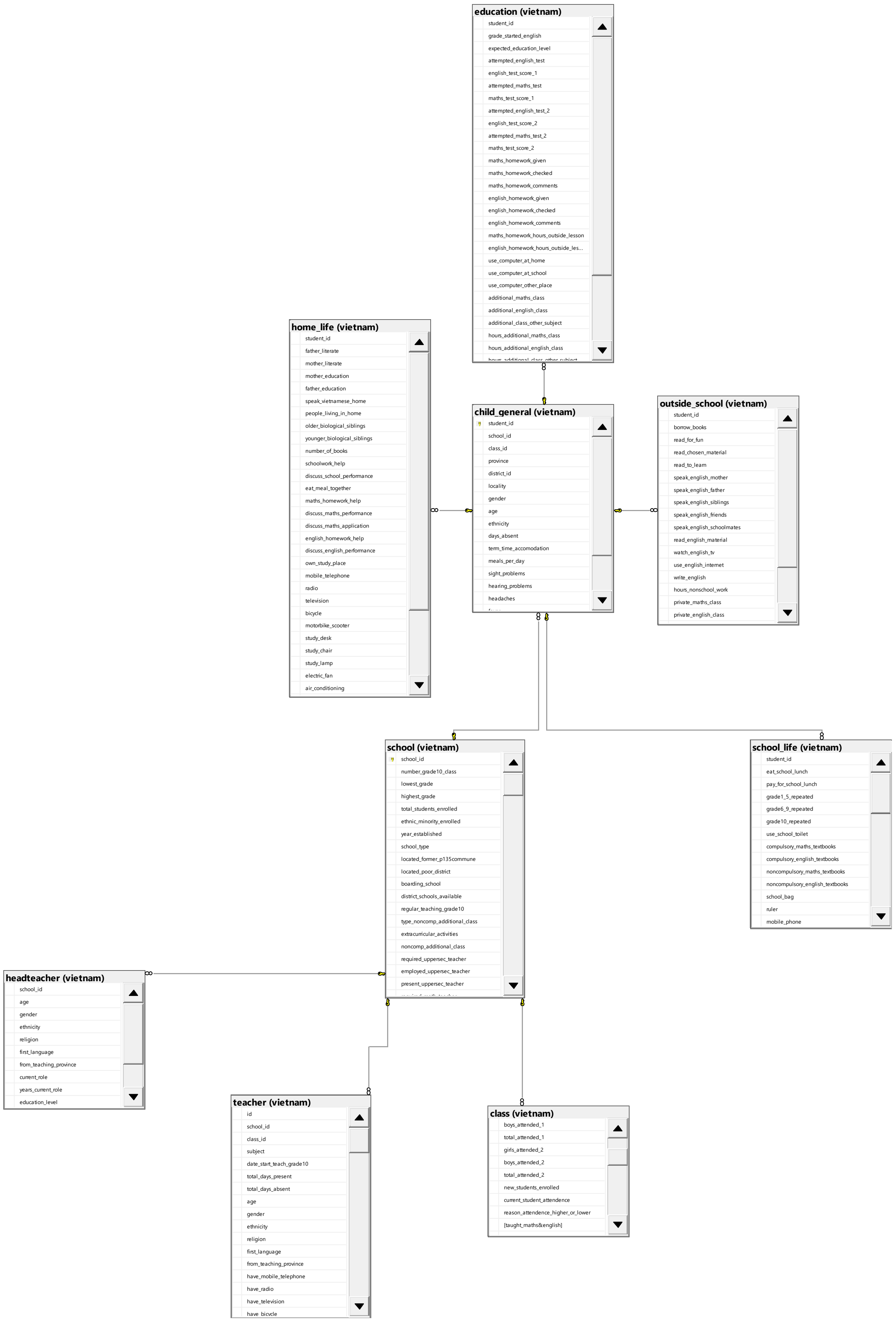
The development of databases has overcome many of these drawbacks. A database is a collection of related data which meets the needs of the organisation (Connolly and Begg, 2015). They provide a centralised location for all relevant and useful data, in the same format, allowing users to efficiently access required data. As related data is linked within the database, users are able to access combinations of data from different tables as required without the need for custom files and programs to be created (Connolly and Begg, 2015). The Database Management System is the software which allows users to interact with the stored data. They also provide database security, protect data integrity, create a concurrency control system so multiple users can access the system at one time, and a recovery system which reverts the database back to a previous consistent state in the event of failure, preventing data loss (Connolly and Begg, 2015). SQL Server is a popular DBMS software developed by Microsoft, which offers all of these services. Overall, databases allow organisations to speed up their analysis and therefore decision making process, and allow organisations to collect more data than they previously would have been able to (Connolly and Begg, 2015, Elmasri and Navathe, 2017).

The Young Lives project conducted a School Survey to investigate inequality in the education sector in Vietnam between 2016 and 2017. This involved providing children, their families and schools a questionnaire at the beginning of the school year (wave 1) and the end of the school year (wave 2), which measured a variety of topics relating to education, health and wealth (Iyer et al., 2017). The organisation would benefit from a database system to store collected data efficiently and build reports to analyse education inequality. This task will use SQL Server 2018 to import the recorded data into a database, organise the data into logically related tables and a summary report via Microsoft Excel that users could use to analyse education inequality in Vietnam.

2. Design Rationale

SQL is an example of a Relational Database Management System, which is based on the relational data model proposed by Codd (1972). All data is represented in tables (relations) with columns (attributes) and rows (tuple) which contain one value per attribute. The relational model must meet a set of objectives: allow data independence, and allow for normalisation which deals with problems relating to data semantics, consistency and redundancy (Connolly and Begg, 2015). To ensure these objectives are met, relations and attributes must have a distinct name, attribute values must be the same type, tuples must be distinct, each attribute contains a single value per tuple, and the order of tuples and attributes must not be significant (Connolly and Begg, 2015, Elmasri and Navathe, 2017). There are a number of design considerations which help to ensure these requirements are met, including normalisation of relations, and setting constraints, which will be discussed in detail later.

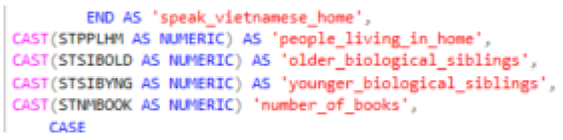
Before database implementation, conceptual design of the database is essential in order for the database to meet the goals of the users. Understanding the overall business goals and the users requirements of the database allows the selection of appropriate data and how this data should be organised within the database (Connolly and Begg, 2015). The School Survey study has already gathered the required data to meet their goals. The aim of the required database is storing the data from both waves appropriately in a way which facilitates user investigation into education inequality in Vietnam. For the conceptual design of the database, it is popular to follow the entity-relationship (ER) model. The first step is to identify entities and their attributes. Entities are ‘objects’ with their own independent existence and can be either physical or abstract (Connolly and Begg, 2015). Each entity then has corresponding attributes which describe the properties of interest to the users (Elmasri and Navathe, 2017). In this instance, there are two entities: each child can be considered an entity, with details of their education, school life and home life the attributes, and the school another entity, with the class, headteacher, school and teacher details being the attributes for this. Most of the attributes can be considered to be single-valued attributes as they each child’s response is recorded once for each wave, with the waves recording different details. Some of the surveys were not filled in, such as the child not being present for the second wave survey and teachers/headteachers not filling in their survey. Any data missing data for each child and school needs to be filled in with a NULL value. However, one attribute must not have any NULL values and this is the key attribute (Connolly and Begg, 2015). The primary key attribute is distinct for each entity so can be used to identify different entities. In the School Survey study, each child and school were given a unique identifier. These will be used as the key attributes. Often in databases there are multiple entities which relate to each other and so can be linked together via these relationships. Additionally, it is important to consider minimising redundancy within the database model (Connolly and Begg, 2015). Therefore, the relation was broken down further to group measurements which fulfilled the same purpose and linked by the relationships to the primary keys. For example, the child’s general information, the child’s education, the child’s home life, the child’s life outside of the class, the school, the child’s school life were all linked by the child’s ID. The teacher details, the school details, the headteacher details were linked via the school’s ID.


education (vietnam)
student_id
grade_started_english
expected_education_level
attempted_english_test
english_test_score_1
attempted_maths_test
maths_test_score_1
attempted_english_test_2
english_test_score_2
attempted_maths_test_2
maths_test_score_2
maths_homework_given
maths_homework_checked
maths_homework_comments
english_homework_given
english_homework_checked
english_homework_comments
maths_homework_hours_outside_lesson
english_homework_hours_outside_les...
use_computer_at_home
use_computer_at_school
use_computer_other_place
additional_maths_class
additional_english_class
additional_class_other_subject
hours_additional_maths_class
hours_additional_english_class
home_life (vietnam)
student_id
father_literate
mother_literate
mother_education
father_education
speak_vietnamese_home
people_living_in_home
older_biological_siblings
younger_biological_siblings
number_of_books
schoolwork_help
discuss_school_performance
eat_meal_together
maths_homework_help
discuss_maths_performance
discuss_maths_application
english_homework_help
discuss_english_performance
own_study_place
mobile_telephone
radio
television
bicycle
motorbike_scooter
study_desk
study_chair
study_lamp
electric_fan
air_conditioning
child_general (vietnam)
student_id
school_id
class_id
province
district_id
locality
gender
age
ethnicity
days_absent
term_time_accomodation
meals_per_day
sight_problems
hearing_problems
headaches
outside_school (vietnam)
student_id
borrow_books
read_for_fun
read_chosen_material
read_to_learn
speak_english_mother
speak_english_father
speak_english_siblings
speak_english_friends
speak_english_schoolmates
read_english_material
watch_english_tv
use_english_internet
write_english
hours_nonschool_work
private_maths_class
private_english_class
school (vietnam)
school_id
number_grade10_class
lowest_grade
highest_grade
total_students_enrolled
ethnic_minority_enrolled
year_established
school_type
located_former_p135commune
located_poor_district
boarding_school
district_schools_available
regular_teaching_grade10
type_noncomp_additional_class
extracurricular_activities
noncomp_additional_class
required_uppersec_teacher
employed_uppersec_teacher
present_uppersec_teacher
school_life (vietnam)
student_id
eat_school_lunch
pay_for_school_lunch
grade1_5_repeated
grade6_9_repeated
grade10_repeated
use_school_toilet
compulsory_maths_textbooks
compulsory_english_textbooks
noncompulsory_maths_textbooks
noncompulsory_english_textbooks
school_bag
ruler
mobile_phone
headteacher (vietnam)
school_id
age
gender
ethnicity
religion
first_language
from_teaching_province
current_role
years_current_role
education_level
teacher (vietnam)
id
school_id
class_id
subject
date_start_teach_grade10
total_days_present
total_days_absent
age
gender
ethnicity
religion
first_language
from_teaching_province
have_mobile_telephone
have_radio
have_television
have_bicycle
class (vietnam)
boys_attended_1
total_attended_1
girls_attended_2
boys_attended_2
total_attended_2
new_students_enrolled
current_student_attendence
reason_attendence_higher_or_lower
[taught_maths&english]

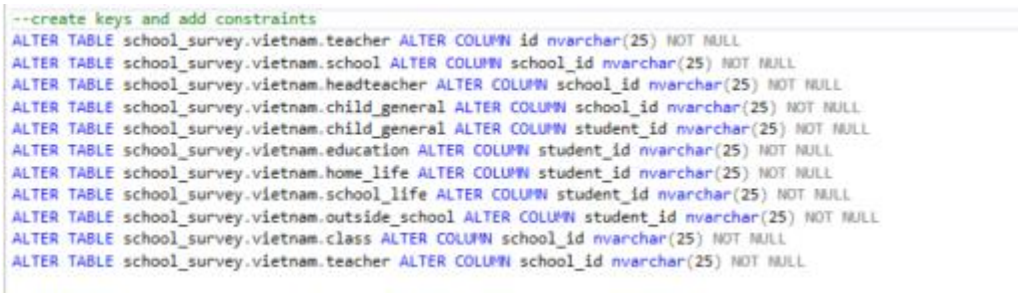
3. Design Considerations

The next stage is to consider the logical database design. This translates the conceptual model into a logical model where the model structure and ability to manage user requirements are evaluated. The stages of developing a logical data model include, normalisation, creating integrity constraints, check transactions can be handled by the model and that the model is secure (Connolly and Begg, 2015),

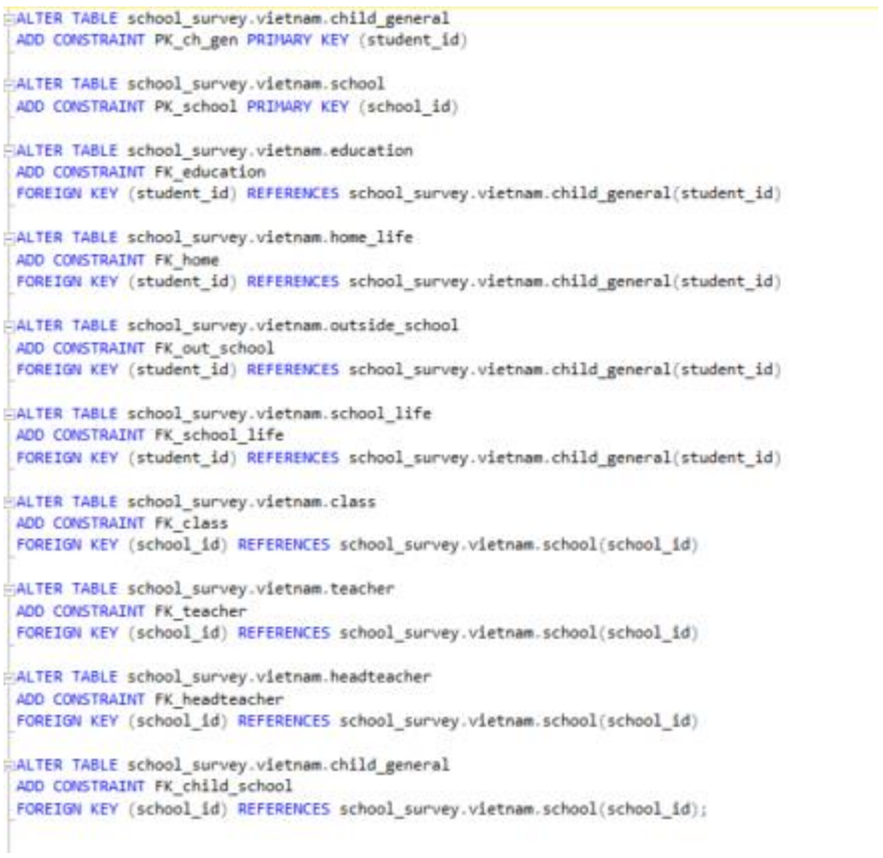
Constraints are necessary to consider in order to ensure data integrity is maintained. These are rules which prevent altering of information in the database which would impact the accuracy or quality of information stored. Constraints can be domain constraints, entity integrity constraints or referential integrity constraints (Elmasri and Navathe, 2017). Domain constraints involve setting column data types. In this instance, columns such as the number of people living with the child at home (‘people_living_at_home’) and the number of books in the house (‘number_of_books’) were cast as numeric. This ensures that strings cannot be inputted into these columns without causing an error. Additionally, numeric values which are considered categorical, such as the class ID (‘class_id’), are set as varchar datatypes to prevent calculations from being performed on these datatypes where would not be applicable. Entity integrity constraints involve preventing primary key values from being null, which in this instance is the child’s ID and the school’s ID. Referential integrity constraints involve all foreign keys in a table matching up to the primary keys in another table.



```
        END AS 'speak_vietnamese_home',
CAST(STPPLHM AS NUMERIC) AS 'people_living_in_home',
CAST(STSIBOLD AS NUMERIC) AS 'older_biological_siblings',
CAST(STSIBYNG AS NUMERIC) AS 'younger_biological_siblings',
CAST(STNMBOOK AS NUMERIC) 'number_of_books',
    CASE
```



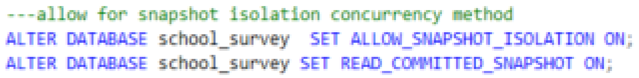
```
--create keys and add constraints
ALTER TABLE school_survey.vietnam.teacher ALTER COLUMN id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.school ALTER COLUMN school_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.headteacher ALTER COLUMN school_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.child_general ALTER COLUMN school_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.child_general ALTER COLUMN student_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.education ALTER COLUMN student_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.home_life ALTER COLUMN student_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.school_life ALTER COLUMN student_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.outside_school ALTER COLUMN student_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.class ALTER COLUMN school_id nvarchar(25) NOT NULL
ALTER TABLE school_survey.vietnam.teacher ALTER COLUMN school_id nvarchar(25) NOT NULL
```



```
ALTER TABLE school_survey.vietnam.child_general
ADD CONSTRAINT PK_ch_gen PRIMARY KEY (student_id)

ALTER TABLE school_survey.vietnam.school
ADD CONSTRAINT PK_school PRIMARY KEY (school_id)

ALTER TABLE school_survey.vietnam.education
ADD CONSTRAINT FK_education
FOREIGN KEY (student_id) REFERENCES school_survey.vietnam.child_general(student_id)

ALTER TABLE school_survey.vietnam.home_life
ADD CONSTRAINT FK_home
FOREIGN KEY (student_id) REFERENCES school_survey.vietnam.child_general(student_id)

ALTER TABLE school_survey.vietnam.outside_school
ADD CONSTRAINT FK_out_school
FOREIGN KEY (student_id) REFERENCES school_survey.vietnam.child_general(student_id)

ALTER TABLE school_survey.vietnam.school_life
ADD CONSTRAINT FK_school_life
FOREIGN KEY (student_id) REFERENCES school_survey.vietnam.child_general(student_id)

ALTER TABLE school_survey.vietnam.class
ADD CONSTRAINT FK_class
FOREIGN KEY (school_id) REFERENCES school_survey.vietnam.school(school_id)

ALTER TABLE school_survey.vietnam.teacher
ADD CONSTRAINT FK_teacher
FOREIGN KEY (school_id) REFERENCES school_survey.vietnam.school(school_id)

ALTER TABLE school_survey.vietnam.headteacher
ADD CONSTRAINT FK_headteacher
FOREIGN KEY (school_id) REFERENCES school_survey.vietnam.school(school_id)

ALTER TABLE school_survey.vietnam.child_general
ADD CONSTRAINT FK_child_school
FOREIGN KEY (school_id) REFERENCES school_survey.vietnam.school(school_id);
```

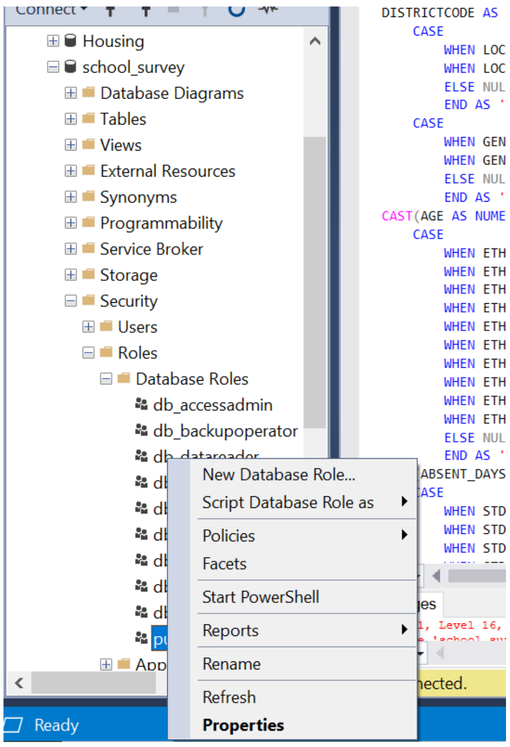
In terms of relational databases, only the first form of normalization is essential (Connolly and Begg, 2015). First normal form requires each cell to contain a single value, each column contains the same type of entry and rows need to be unique. The CSV files containing the recorded data for each wave already follow this form. Each child is given a unique identifier to identify their responses and the child's data for each round is recorded in separate rows. The use of unique identifiers and round numbers ensures that each row is unique. Additionally, each column only contains a set value which was determined as part of the study's design. Additional normal forms are optional (Connolly and Begg, 2015). Second normal form requires all attributes which are not key columns to be dependent on the key. In this instance there are clearly two entities where attributes describe the child or attributes describe the school, so need to be separated. This also meets the third normal form requirement of all columns can only be determined by the table key. Additionally, running queries can sometimes produce errors which prevent them from continuing to run or produce unexpected results (Mukherjee, 2019). To raise awareness for any errors occurring, many sections of code include 'TRY… CATCH' statements which will print error messages should an error occur, allowing administrator intervention.

In multi-user systems, many different users can access the database at the same time. This can create issues with database concurrency if two transactions (units of work) run at the same time (Connolly and Begg, 2015, Elmasri and Navathe, 2017). For example, the lost update problem is caused when two transactions are interleaved, and one transaction reads the data just before another transaction updates the data. This means the transaction will contain old data and so if this transaction also updates the data, any changes made by the previous will be lost. Another issue involves the temporary update (dirty read) problem, where an update made fails and so requires rolling back to the previous data but prior to rolling back this updated data is read by another transaction. The incorrect summary problem involves aggregating transactions obtaining a mixture of updated data and yet-to-be updated data, so is neither an accurate summary of the old data or the updated data. The unrepeatable read problem is caused when a transaction conducts two reads with another transaction updating the value in between, so the transaction reads two different values. Overall, this leads to loss of data integrity and validity. A number of algorithms have been developed to allow transaction concurrency to occur without compromising the data, each with their own strengths and limitations. The method chosen for this database is snapshot isolation, which is a form of optimistic concurrency control. This method involves taking a ‘snapshot’ of the data at the start of the transaction which transactions are conducted on rather than the actual data and once finished any updates can be applied. This method allows reads to occur without needing read ‘locks’, which only allow one read transaction to look at the data at one time, so all read transactions can occur simultaneously (Elmasri and Navathe, 2017). Any transactions which are making any changes to the data will still require locks. This system is useful for this database as the data has already been pre-recorded for the study so should require further editing or updating. It is likely that users will just be using read transactions. In SQL, this concurrency control method is activated within the database using the code in the figure below (Mukherjee, 2019). Transactions will either be committed if the query is successful or rolled back so changes are not committed to the table.



```
---allow for snapshot isolation concurrency method
ALTER DATABASE school_survey  SET ALLOW_SNAPSHOT_ISOLATION ON;
ALTER DATABASE school_survey SET READ_COMMITTED_SNAPSHOT ON;
```

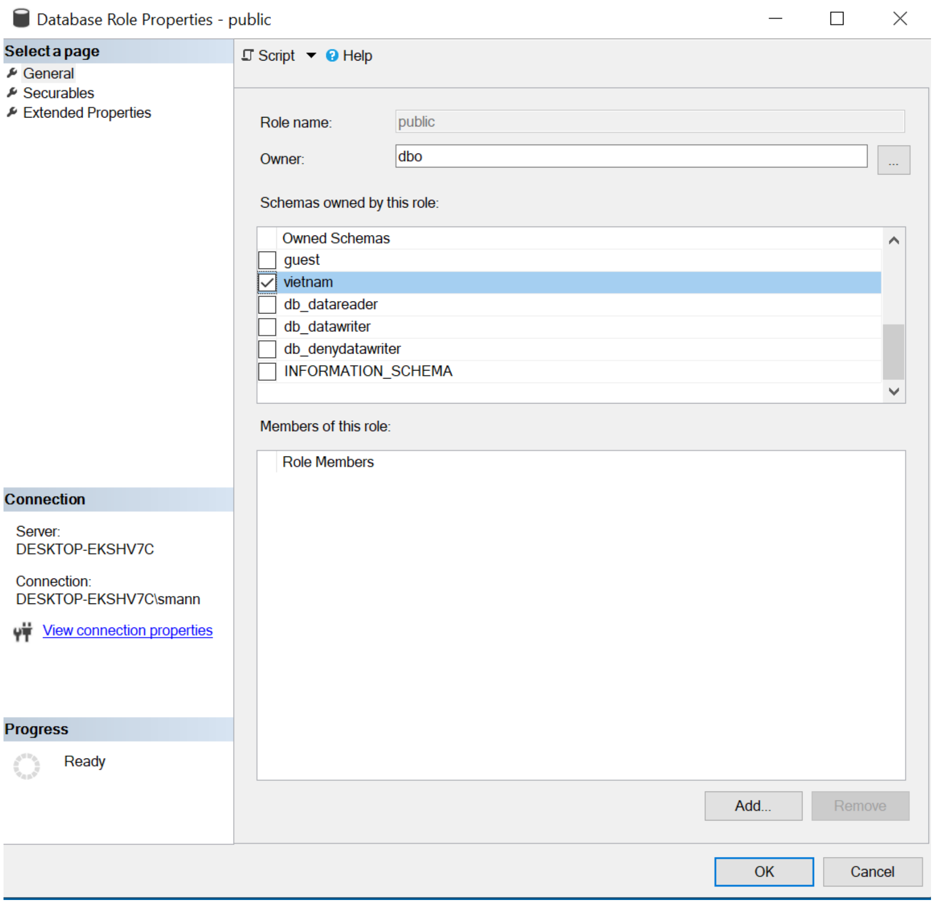
Many users are expected to be able to use the database to access data, which can put the data at risk of damage (whether intentional or unintentional) or leaking of protected information. Fortunately in the case of this study any private details regarding the children have been anonymised (Briones, 2018). There is still the risk of damage to the data held in the database which would impact the ability to use the data for its intended purpose or for data to be leaked to the outside and used without permission by others. Restricting the database access for individuals in a way which they are can only access what is required to complete their task helps to secure the database by reducing the number of individuals with complete access (Connolly and Begg, 2015). Within the database, users can be created and attached to a particular login. There are two options for users to login to the SQL database, using SQL server login or Windows authentication. It is recommended by Microsoft to use Windows authentication as this is more secure (Mukherjee, 2019). Users can then be given particular privileges which restrict them from accessing all areas of the database and restrict what they are able to do within the database, such as being able to update or change the data in any way. If there are many users, it is probably better to assign users to a role where anyone with the same role has the same privileges. All users are automatically assigned the public role, so this should be the lowest level of access a user is awarded. The public role for this database was only given access to the Vietnam schema and was not allowed to view/update/alter tables. Views are an excellent method of security, as they only provide individuals access to the attributes required for them to do their job, so any sensitive information in the tables can be withheld (Connolly and Begg, 2015). Therefore, views were used to select data which users would need to investigate certain aspects of education inequality in Vietnam, without giving them access to all the tables.



```
DISTRICTCODE AS
    CASE
        WHEN LOC
        WHEN LOC
        ELSE NUL
        END AS '
    CASE
        WHEN GEN
        WHEN GEN
        ELSE NUL
        END AS '
CAST(AGE AS NUME
    CASE
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        WHEN ETH
        ELSE NUL
        END AS '
ABSENT_DAYS
ASE
        WHEN STD
        WHEN STD
        WHEN STD
```

```
1, Level 16,
```

ected.


Database Role Properties - public

Role name: public

Owner: dbo

Schemas owned by this role:

Owned Schemas
☐ guest
✓ vietnam
☐ db_datareader
☐ db_datawriter
☐ db_denydatawriter
☐ INFORMATION_SCHEMA

Members of this role:

Role Members

Connection

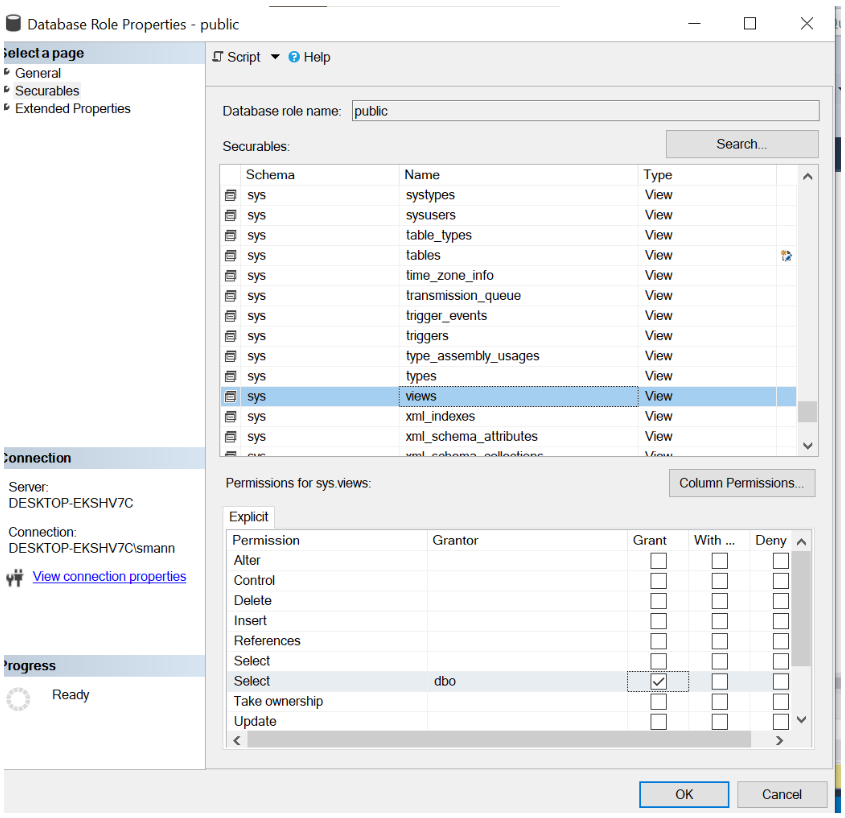
Server:
DESKTOP-EKSHV7C

Connection:
DESKTOP-EKSHV7C\smann

View connection properties

Progress

Ready


Database Role Properties - public

Select a page
- General
- Securables
- Extended Properties

Script ▾ Help

Database role name: public

Securables: Search...

Schema	Name	Type
sys	systypes	View
sys	sysusers	View
sys	table_types	View
sys	tables	View
sys	time_zone_info	View
sys	transmission_queue	View
sys	trigger_events	View
sys	triggers	View
sys	type_assembly_usages	View
sys	types	View
sys	views	View
sys	xml_indexes	View
sys	xml_schema_attributes	View

Connection

Server:
DESKTOP-EKSHV7C

Connection:
DESKTOP-EKSHV7C\smann

View connection properties

Progress

Ready

Permissions for sys.views: Column Permissions...

Explicit

Permission	Grantor	Grant	With ...	Deny
Alter		☐	☐	☐
Control		☐	☐	☐
Delete		☐	☐	☐
Insert		☐	☐	☐
References		☐	☐	☐
Select		☐	☐	☐
Select	dbo	✓	☐	☐
Take ownership		☐	☐	☐
Update		☐	☐	☐

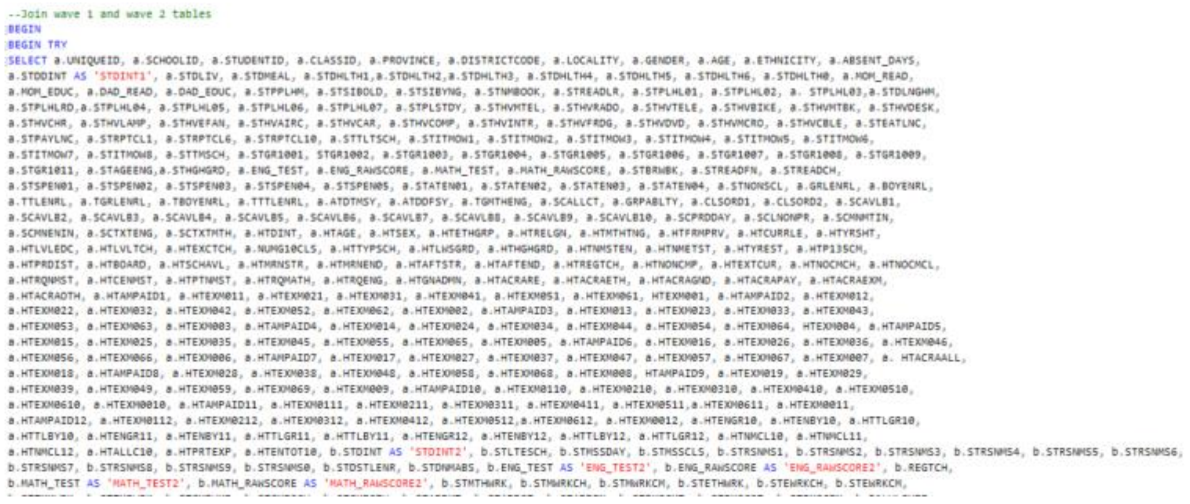
OK Cancel

As multiple users within a database may have privileges which allow them to make edits to the database, including inserting data, creating new functions or stored procedures, etc., it is important to make comments within SQL code to explain the function of sections of code and to explain any changes made to sections of code (Mukherjee, 2019). Additionally, it can be useful to block out sections of code which do not need to be run but may be required in a later instance. Although not applied here, it is good convention to mark any changes to code with comments to explain the changes, the date of change and who created the change, particularly if there are multiple users with the privileges to make database changes.

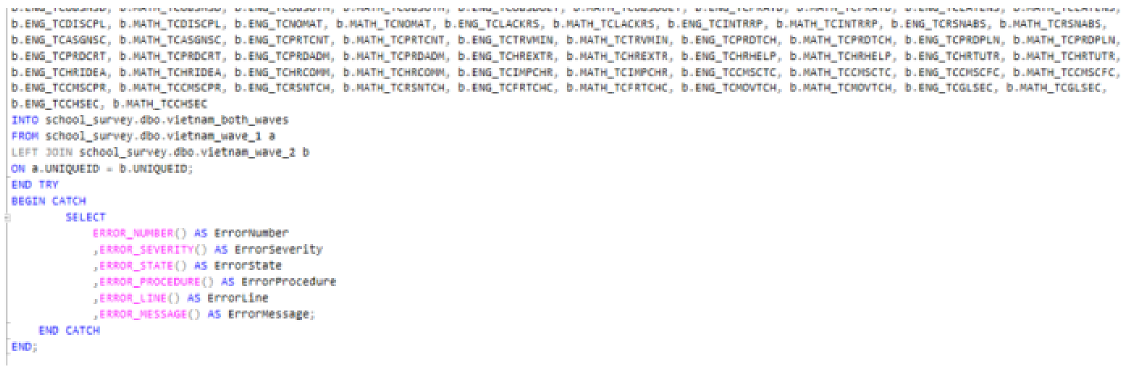
4. T-SQL statements

The CSV files for each wave were imported into the 'school_survey' database using SSMS's Import and Export Wizard, which generates a table automatically, with the first row used as the column attributes and the delimiter set used to separate data into appropriate columns. Following this the two waves were joined together using 'LEFT JOIN' using the child's unique identifier, creating a table called 'vietnam_both_waves'.

```
---create school_survey database
USE master;
GO
CREATE DATABASE school_survey;
```

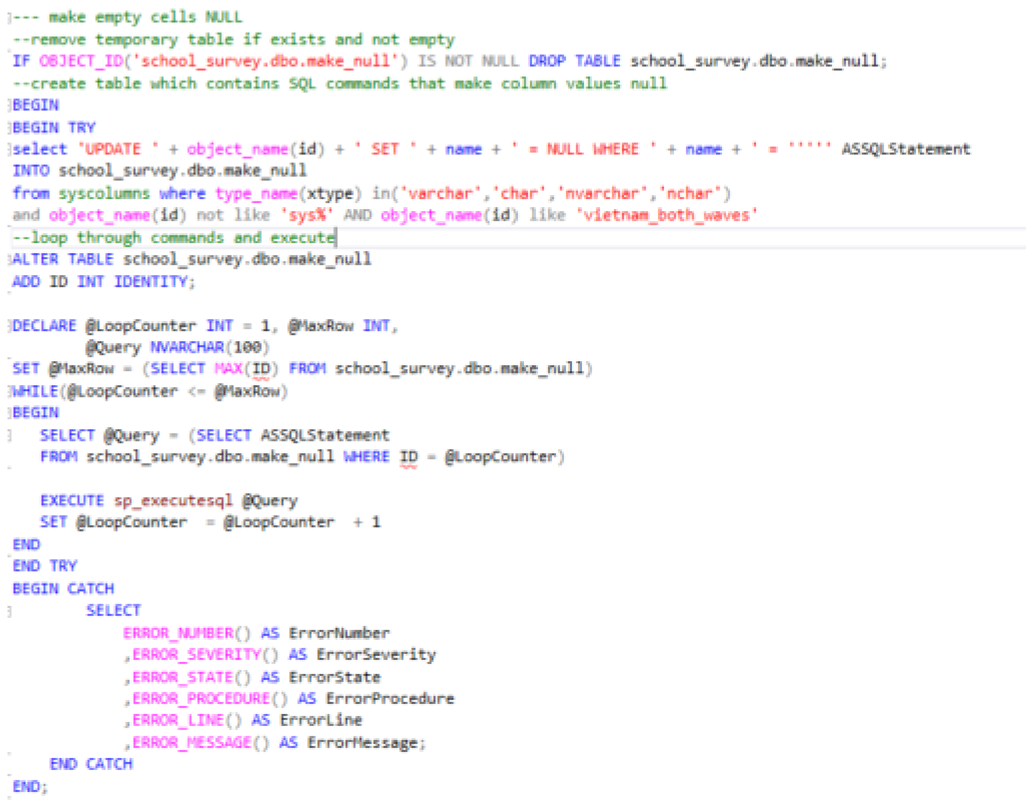


```
--Join wave 1 and wave 2 tables
BEGIN
BEGIN TRY
SELECT a.UNIQUEID, a.SCHOOLID, a.STUDENTID, a.CLASSID, a.PROVINCE, a.DISTRICTCODE, a.LOCALITY, a.GENDER, a.AGE, a.ETHNICITY, a.ABSENT_DAYS,
a.STDDINT AS 'STDINT1', a.STDLIV, a.STDMEAL, a.STDHLTH1,a.STDHLTH2,a.STDHLTH3, a.STDHLTH4, a.STDHLTH5, a.STDHLTH6, a.STDHLTH0, a.MOM_READ,
a.MOM_EDUC, a.DAD_READ, a.DAD_EDUC, a.STPPLHM, a.STSIBOLD, a.STSIBYNG, a.STNMBOOK, a.STREADLR, a.STPLHL01, a.STPLHL02, a. STPLHL03,a.STDLNGHM,
a.STPLHLRD,a.STPLHL04, a.STPLHL05, a.STPLHL06, a.STPLHL07, a.STPLSTDY, a.STHVMTEL, a.STHVRADO, a.STHVTELE, a.STHVBIKE, a.STHVMTBK, a.STHVDESK,
a.STHVCHR, a.STHVLAMP, a.STHVEFAN, a.STHVAIRC, a.STHVCAR, a.STHVCOMP, a.STHVINTR, a.STHVFRDG, a.STHVDVD, a.STHVMCRO, a.STHVCBLE, a.STEATLNC,
a.STPAYLNC, a.STRPTCL1, a.STRPTCL6, a.STRPTCL10, a.STTLTSCH, a.STITMOW1, a.STITMOW2, a.STITMOW3, a.STITMOW4, a.STITMOW5, a.STITMOW6,
a.STITMOW7, a.STITMOW8, a.STTMSCH, a.STGR1001, STGR1002, a.STGR1003, a.STGR1004, a.STGR1005, a.STGR1006, a.STGR1007, a.STGR1008, a.STGR1009,
a.STGR1011, a.STAGEENG,a.STHGHGRD, a.ENG_TEST, a.ENG_RAWSCORE, a.MATH_TEST, a.MATH_RAWSCORE, a.STBRWBK, a.STREADFN, a.STREADCH,
a.STSPEN01, a.STSPEN02, a.STSPEN03, a.STSPEN04, a.STSPEN05, a.STATEN01, a.STATEN02, a.STATEN03, a.STATEN04, a.STNONSCL, a.GRLENRL, a.BOYENRL,
a.TTLENRL, a.TGRLENRL, a.TBOYENRL, a.TTTLENRL, a.ATDTMSY, a.ATDDFSY, a.TGMTHENG, a.SCALLCT, a.GRPABLTY, a.CLSORD1, a.CLSORD2, a.SCAVLB1,
a.SCAVLB2, a.SCAVLB3, a.SCAVLB4, a.SCAVLB5, a.SCAVLB6, a.SCAVLB7, a.SCAVLB8, a.SCAVLB9, a.SCAVLB10, a.SCPRDDAY, a.SCLNONPR, a.SCMNMTIN,
a.SCMNENIN, a.SCTXTENG, a.SCTXTMTH, a.HTDINT, a.HTAGE, a.HTSEX, a.HTETHGRP, a.HTRELGN, a.HTMTHTNG, a.HTFRMPRV, a.HTCURRLE, a.HTYRSHT,
a.HTLVLEDC, a.HTLVLTCH, a.HTEXCTCH, a.NUMG10CLS, a.HTTYPSCH, a.HTLWSGRD, a.HTHGHGRD, a.HTNMSTEN, a.HTNMETST, a.HTYREST, a.HTP135CM,
a.HTPRDIST, a.HTBOARD, a.HTSCHAVL, a.HTMRNSTR, a.HTMRNEND, a.HTAFTSTR, a.HTAFTEND, a.HTREGTCH, a.HTNONCMP, a.HTEXTCUR, a.HTNOCMCH, a.HTNOCMCL,
a.HTRQNMST, a.HTCENMST, a.HTPTNMST, a.HTRQMATH, a.HTRQENG, a.HTGNADMN, a.HTACRARE, a.HTACRAETH, a.HTACRAGND, a.HTACRAPAY, a.HTACRAEXM,
a.HTACRAOTH, a.HTAMPAID1, a.HTEXM011, a.HTEXM021, a.HTEXM031, a.HTEXM041, a.HTEXM051, a.HTEXM061, HTEXM001, a.HTAMPAID2, a.HTEXM012,
a.HTEXM022, a.HTEXM032, a.HTEXM042, a.HTEXM052, a.HTEXM062, a.HTEXM002, a.HTAMPAID3, a.HTEXM013, a.HTEXM023, a.HTEXM033, a.HTEXM043,
a.HTEXM053, a.HTEXM063, a.HTEXM003, a.HTAMPAID4, a.HTEXM014, a.HTEXM024, a.HTEXM034, a.HTEXM044, a.HTEXM054, a.HTEXM064, HTEXM004, a.HTAMPAID5,
a.HTEXM015, a.HTEXM025, a.HTEXM035, a.HTEXM045, a.HTEXM055, a.HTEXM065, a.HTEXM005, a.HTAMPAID6, a.HTEXM016, a.HTEXM026, a.HTEXM036, a.HTEXM046,
a.HTEXM056, a.HTEXM066, a.HTEXM006, a.HTAMPAID7, a.HTEXM017, a.HTEXM027, a.HTEXM037, a.HTEXM047, a.HTEXM057, a.HTEXM067, a.HTEXM007, a. HTACRAALL,
a.HTEXM018, a.HTAMPAID8, a.HTEXM028, a.HTEXM038, a.HTEXM048, a.HTEXM058, a.HTEXM068, a.HTEXM008, HTAMPAID9, a.HTEXM019, a.HTEXM029,
a.HTEXM039, a.HTEXM049, a.HTEXM059, a.HTEXM069, a.HTEXM009, a.HTAMPAID10, a.HTEXM0110, a.HTEXM0210, a.HTEXM0310, a.HTEXM0410, a.HTEXM0510,
a.HTEXM0610, a.HTEXM0010, a.HTAMPAID11, a.HTEXM0111, a.HTEXM0211, a.HTEXM0311, a.HTEXM0411, a.HTEXM0511,a.HTEXM0611, a.HTEXM0011,
a.HTAMPAID12, a.HTEXM0112, a.HTEXM0212, a.HTEXM0312, a.HTEXM0412, a.HTEXM0512,a.HTEXM0612, a.HTEXM0012, a.HTENGR10, a.HTENBY10, a.HTTLGR10,
a.HTTLBY10, a.HTENGR11, a.HTENBY11, a.HTTLGR11, a.HTTLBY11, a.HTENGR12, a.HTENBY12, a.HTTLBY12, a.HTTLGR12, a.HTNMCL10, a.HTNMCL11,
a.HTNMCL12, a.HTALLC10, a.HTPRTEXP, a.HTENTOT10, b.STDINT AS 'STDINT2', b.STLTESCH, b.STMSSDAY, b.STMSSCLS, b.STRSNMS1, b.STRSNMS2, b.STRSNMS3, b.STRSNMS4, b.STRSNMS5, b.STRSNMS6,
b.STRSNMS7, b.STRSNMS8, b.STRSNMS9, b.STRSNMS0, b.STDSTLENR, b.STDNMABS, b.ENG_TEST AS 'ENG_TEST2', b.ENG_RAWSCORE AS 'ENG_RAWSCORE2', b.REGTCH,
b.MATH_TEST AS 'MATH_TEST2', b.MATH_RAWSCORE AS 'MATH_RAWSCORE2', b.STMTHWRK, b.STMWRKCH, b.STMWRKCM, b.STETHWRK, b.STEWRKCH, b.STEWRKCM,
```



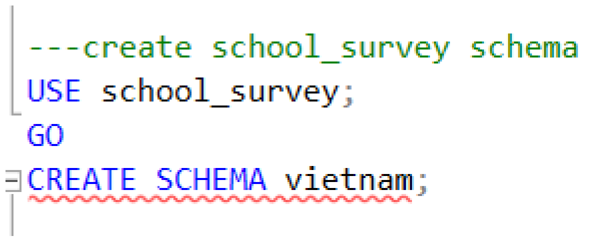
```
b.ENG_TCDISCPL, b.MATH_TCDISCPL, b.ENG_TCNOMAT, b.MATH_TCNOMAT, b.ENG_TCLACKRS, b.MATH_TCLACKRS, b.ENG_TCINTRRP, b.MATH_TCINTRRP, b.ENG_TCRSNABS, b.MATH_TCRSNABS,
b.ENG_TCASGNSC, b.MATH_TCASGNSC, b.ENG_TCPRTCNT, b.MATH_TCPRTCNT, b.ENG_TCTRVMIN, b.MATH_TCTRVMIN, b.ENG_TCPRDTCH, b.MATH_TCPRDTCH, b.ENG_TCPRDPLN, b.MATH_TCPRDPLN,
b.ENG_TCPRDCRT, b.MATH_TCPRDCRT, b.ENG_TCPRDADM, b.MATH_TCPRDADM, b.ENG_TCHREXTR, b.MATH_TCHREXTR, b.ENG_TCHRHELP, b.MATH_TCHRHELP, b.ENG_TCHRTUTR, b.MATH_TCHRTUTR,
b.ENG_TCHRIDEA, b.MATH_TCHRIDEA, b.ENG_TCHRCOMM, b.MATH_TCHRCOMM, b.ENG_TCIMPCHR, b.MATH_TCIMPCHR, b.ENG_TCCMSCTC, b.MATH_TCCMSCTC, b.ENG_TCCMSCFC, b.MATH_TCCMSCFC,
b.ENG_TCCMSCPR, b.MATH_TCCMSCPR, b.ENG_TCRSNTCH, b.MATH_TCRSNTCH, b.ENG_TCFRTCHC, b.MATH_TCFRTCHC, b.ENG_TCMOVTCH, b.MATH_TCMOVTCH, b.ENG_TCGLSEC, b.MATH_TCGLSEC,
b.ENG_TCCHSEC, b.MATH_TCCHSEC
INTO school_survey.dbo.vietnam_both_waves
FROM school_survey.dbo.vietnam_wave_1 a
LEFT JOIN school_survey.dbo.vietnam_wave_2 b
ON a.UNIQUEID = b.UNIQUEID;
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

The data provided was all in numeric format and so needed transforming into meaningful data to allow interpretation when building reports. Prior to this, all columns which did not contain any values needed to be transformed to NULL values. This is because when casting columns with numeric data types, it is not possible when blank values are considered strings and so raises an error.

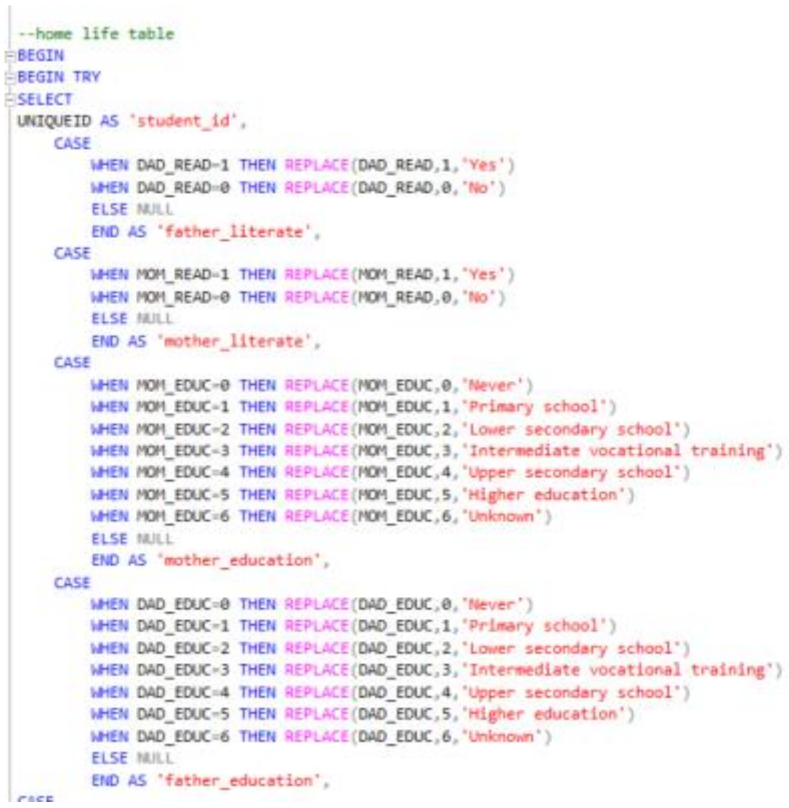


```
--- make empty cells NULL
--remove temporary table if exists and not empty
IF OBJECT_ID('school_survey.dbo.make_null') IS NOT NULL DROP TABLE school_survey.dbo.make_null;
--create table which contains SQL commands that make column values null
BEGIN
BEGIN TRY
select 'UPDATE ' + object_name(id) + ' SET ' + name + ' = NULL WHERE ' + name + ' = ''''' ASSQLStatement
INTO school_survey.dbo.make_null
from syscolumns where type_name(xtype) in('varchar','char','nvarchar','nchar')
and object_name(id) not like 'sys%' AND object_name(id) like 'vietnam_both_waves'
--loop through commands and execute
ALTER TABLE school_survey.dbo.make_null
ADD ID INT IDENTITY;

DECLARE @LoopCounter INT = 1, @MaxRow INT,
        @Query NVARCHAR(100)
SET @MaxRow = (SELECT MAX(ID) FROM school_survey.dbo.make_null)
WHILE(@LoopCounter <= @MaxRow)
BEGIN
   SELECT @Query = (SELECT ASSQLStatement
   FROM school_survey.dbo.make_null WHERE ID = @LoopCounter)

   EXECUTE sp_executesql @Query
   SET @LoopCounter  = @LoopCounter  + 1
END
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

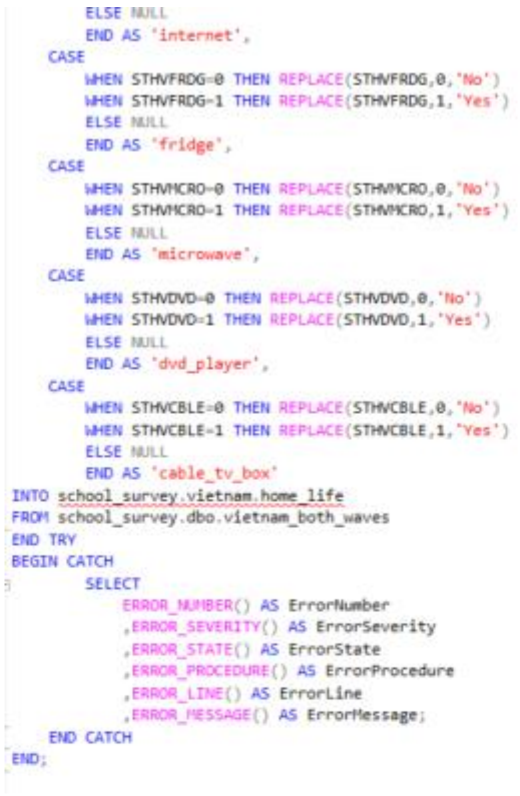
Schema creation within a database can add an extra layer of security by ensuring certain users can only access particular schemas that are relevant to their work. Therefore, when creating the tables with the purpose of measuring education inequality in Vietnam, these were grouped into the newly created schema ‘vietnam’. The data from the table containing all both waves was split into nine tables. Columns for each table were selected and transformed into meaningful data using the commands ‘CASE WHEN’ and then inserted into new tables.



```
---create school_survey schema
USE school_survey;
GO
CREATE SCHEMA vietnam;
```

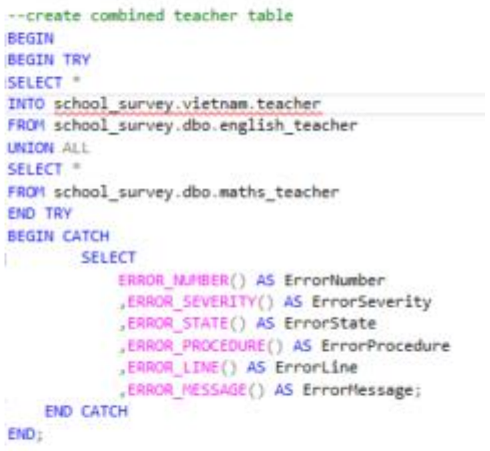


```sql
--home life table
BEGIN
BEGIN TRY
SELECT
UNIQUEID AS 'student_id',
    CASE
        WHEN DAD_READ=1 THEN REPLACE(DAD_READ,1,'Yes')
        WHEN DAD_READ=0 THEN REPLACE(DAD_READ,0,'No')
        ELSE NULL
        END AS 'father_literate',
    CASE
        WHEN MOM_READ=1 THEN REPLACE(MOM_READ,1,'Yes')
        WHEN MOM_READ=0 THEN REPLACE(MOM_READ,0,'No')
        ELSE NULL
        END AS 'mother_literate',
    CASE
        WHEN MOM_EDUC=0 THEN REPLACE(MOM_EDUC,0,'Never')
        WHEN MOM_EDUC=1 THEN REPLACE(MOM_EDUC,1,'Primary school')
        WHEN MOM_EDUC=2 THEN REPLACE(MOM_EDUC,2,'Lower secondary school')
        WHEN MOM_EDUC=3 THEN REPLACE(MOM_EDUC,3,'Intermediate vocational training')
        WHEN MOM_EDUC=4 THEN REPLACE(MOM_EDUC,4,'Upper secondary school')
        WHEN MOM_EDUC=5 THEN REPLACE(MOM_EDUC,5,'Higher education')
        WHEN MOM_EDUC=6 THEN REPLACE(MOM_EDUC,6,'Unknown')
        ELSE NULL
        END AS 'mother_education',
    CASE
        WHEN DAD_EDUC=0 THEN REPLACE(DAD_EDUC,0,'Never')
        WHEN DAD_EDUC=1 THEN REPLACE(DAD_EDUC,1,'Primary school')
        WHEN DAD_EDUC=2 THEN REPLACE(DAD_EDUC,2,'Lower secondary school')
        WHEN DAD_EDUC=3 THEN REPLACE(DAD_EDUC,3,'Intermediate vocational training')
        WHEN DAD_EDUC=4 THEN REPLACE(DAD_EDUC,4,'Upper secondary school')
        WHEN DAD_EDUC=5 THEN REPLACE(DAD_EDUC,5,'Higher education')
        WHEN DAD_EDUC=6 THEN REPLACE(DAD_EDUC,6,'Unknown')
        ELSE NULL
        END AS 'father_education',
```

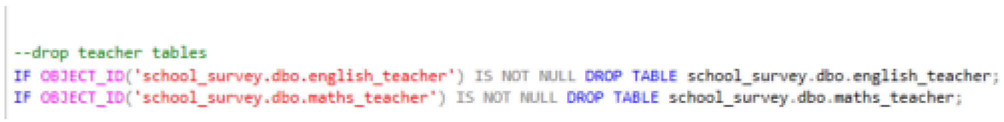


```sql
        ELSE NULL
        END AS 'internet',
    CASE
        WHEN STHVFRDG=0 THEN REPLACE(STHVFRDG,0,'No')
        WHEN STHVFRDG=1 THEN REPLACE(STHVFRDG,1,'Yes')
        ELSE NULL
        END AS 'fridge',
    CASE
        WHEN STHVMCRO=0 THEN REPLACE(STHVMCRO,0,'No')
        WHEN STHVMCRO=1 THEN REPLACE(STHVMCRO,1,'Yes')
        ELSE NULL
        END AS 'microwave',
    CASE
        WHEN STHVDVD=0 THEN REPLACE(STHVDVD,0,'No')
        WHEN STHVDVD=1 THEN REPLACE(STHVDVD,1,'Yes')
        ELSE NULL
        END AS 'dvd_player',
    CASE
        WHEN STHVCBLE=0 THEN REPLACE(STHVCBLE,0,'No')
        WHEN STHVCBLE=1 THEN REPLACE(STHVCBLE,1,'Yes')
        ELSE NULL
        END AS 'cable_tv_box'
INTO school_survey.vietnam.home_life
FROM school_survey.dbo.vietnam_both_waves
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

The responses from the Maths and the English teachers were recorded in separate columns, however in the database these were going to be combined into the same table with the teachers ID and subject being used to distinguish them. To achieve this the maths teacher responses and English teacher responses were inserted into separate tables. Then these tables were appended using ‘UNION ALL’ to create the ‘teacher’ table which contained all teacher responses. The temporary separate teacher tables were then dropped as they were no longer needed.

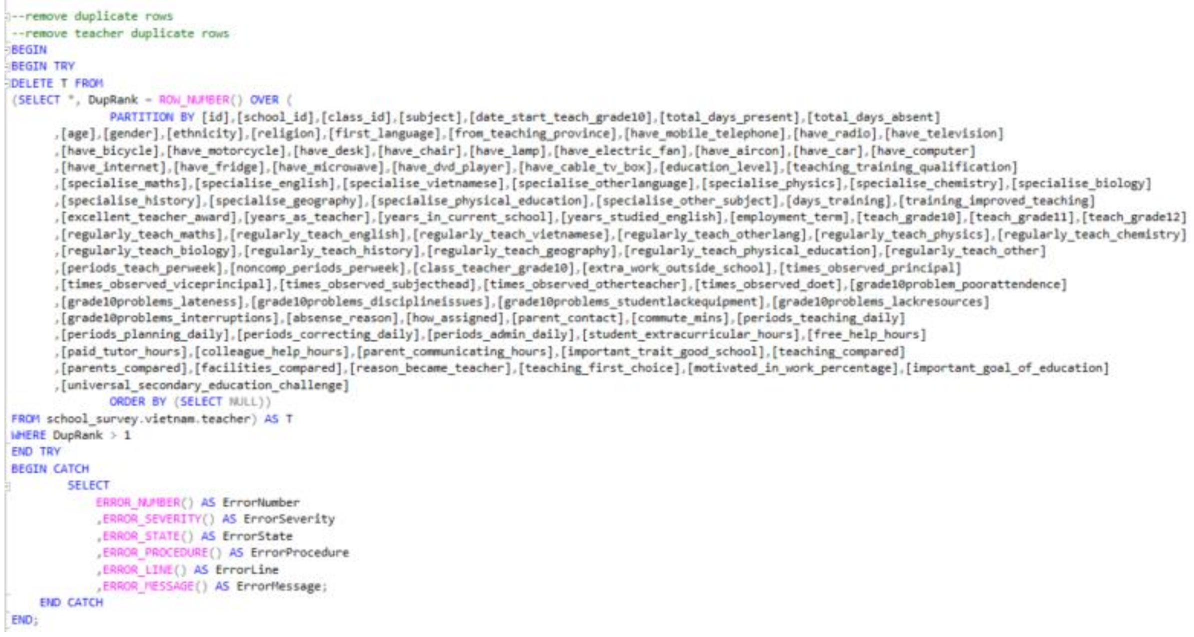


```
--create combined teacher table
BEGIN
BEGIN TRY
SELECT *
INTO school_survey.vietnam.teacher
FROM school_survey.dbo.english_teacher
UNION ALL
SELECT *
FROM school_survey.dbo.maths_teacher
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

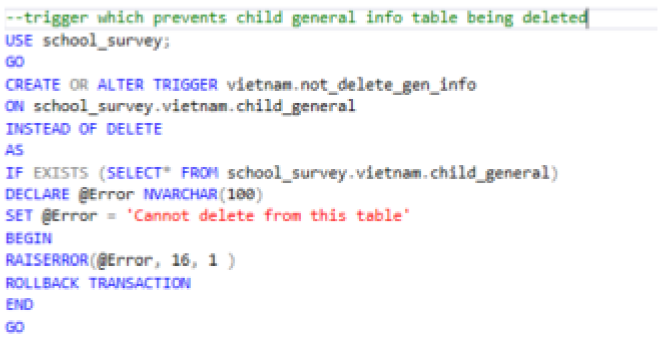


```
--drop teacher tables
IF OBJECT_ID('school_survey.dbo.english_teacher') IS NOT NULL DROP TABLE school_survey.dbo.english_teacher;
IF OBJECT_ID('school_survey.dbo.maths_teacher') IS NOT NULL DROP TABLE school_survey.dbo.maths_teacher;
```

As the initial tables were recorded as survey response for each child, the data which concerned teacher responses, headteacher responses, class details and school details were duplicated for each child. These duplicates were removed once separated into their own tables to remove data redundancy. Additionally, the 'teacher' table contained some NULL rows which were also removed. To ensure data essential data was not lost from the main table ('child_general'), a trigger was created which stops any deletion from occurring on this table and prints an error message. This ensures data is not lost in dependent tables and views, as well as protects data accuracy and integrity by preventing crime reports from being removed.

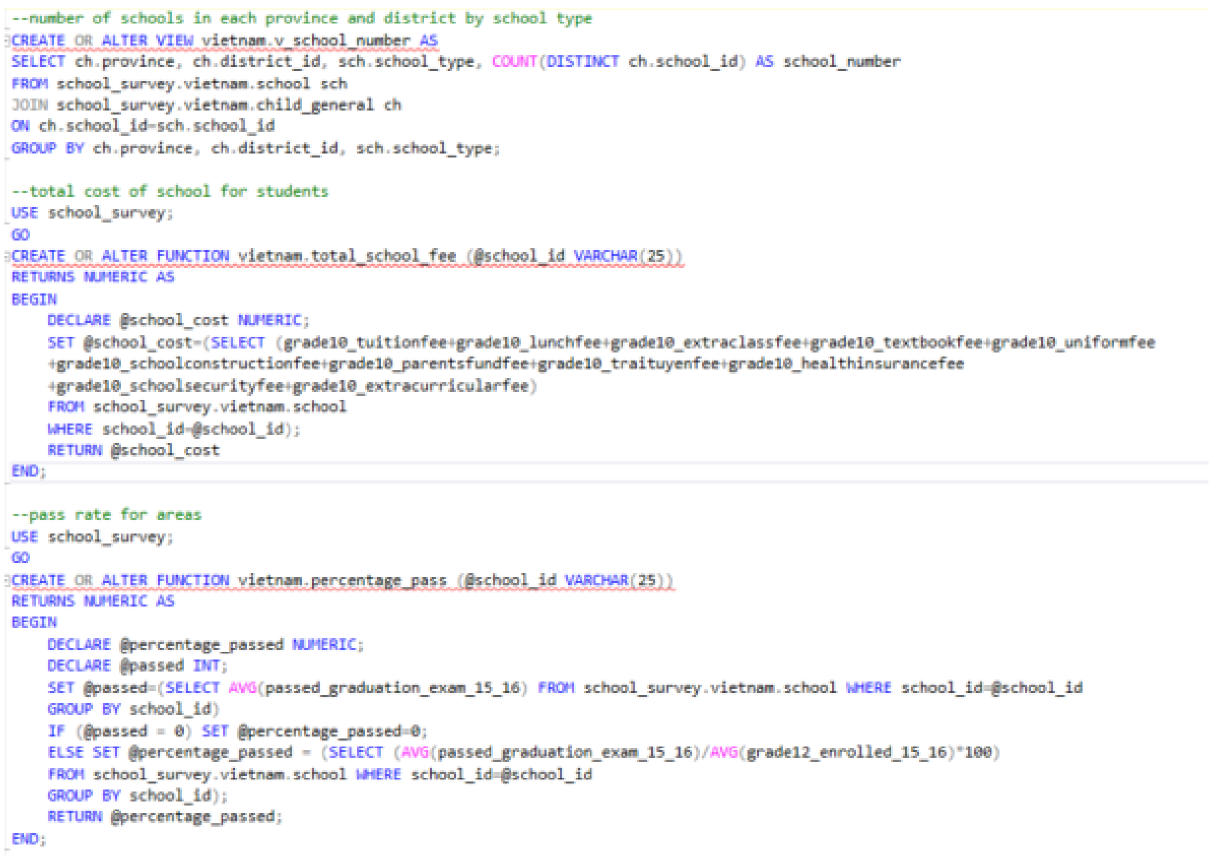


```
--remove duplicate rows
--remove teacher duplicate rows
BEGIN
BEGIN TRY
DELETE T FROM
(SELECT *, DupRank = ROW_NUMBER() OVER (
            PARTITION BY [id],[school_id],[class_id],[subject],[date_start_teach_grade10],[total_days_present],[total_days_absent]
        ,[age],[gender],[ethnicity],[religion],[first_language],[from_teaching_province],[have_mobile_telephone],[have_radio],[have_television]
        ,[have_bicycle],[have_motorcycle],[have_desk],[have_chair],[have_lamp],[have_electric_fan],[have_aircon],[have_car],[have_computer]
        ,[have_internet],[have_fridge],[have_microwave],[have_dvd_player],[have_cable_tv_box],[education_level],[teaching_training_qualification]
        ,[specialise_maths],[specialise_english],[specialise_vietnamese],[specialise_otherlanguage],[specialise_physics],[specialise_chemistry],[specialise_biology]
        ,[specialise_history],[specialise_geography],[specialise_physical_education],[specialise_other_subject],[days_training],[training_improved_teaching]
        ,[excellent_teacher_award],[years_as_teacher],[years_in_current_school],[years_studied_english],[employment_term],[teach_grade10],[teach_grade11],[teach_grade12]
        ,[regularly_teach_maths],[regularly_teach_english],[regularly_teach_vietnamese],[regularly_teach_otherlang],[regularly_teach_physics],[regularly_teach_chemistry]
        ,[regularly_teach_biology],[regularly_teach_history],[regularly_teach_geography],[regularly_teach_physical_education],[regularly_teach_other]
        ,[periods_teach_perweek],[noncomp_periods_perweek],[class_teacher_grade10],[extra_work_outside_school],[times_observed_principal]
        ,[times_observed_viceprincipal],[times_observed_subjecthead],[times_observed_otherteacher],[times_observed_doet],[grade10problem_poorattendence]
        ,[grade10problems_lateness],[grade10problems_disciplineissues],[grade10problems_studentlackequipment],[grade10problems_lackresources]
        ,[grade10problems_interruptions],[absense_reason],[how_assigned],[parent_contact],[commute_mins],[periods_teaching_daily]
        ,[periods_planning_daily],[periods_correcting_daily],[periods_admin_daily],[student_extracurricular_hours],[free_help_hours]
        ,[paid_tutor_hours],[colleague_help_hours],[parent_communicating_hours],[important_trait_good_school],[teaching_compared]
        ,[parents_compared],[facilities_compared],[reason_became_teacher],[teaching_first_choice],[motivated_in_work_percentage],[important_goal_of_education]
        ,[universal_secondary_education_challenge]
            ORDER BY (SELECT NULL))
FROM school_survey.vietnam.teacher) AS T
WHERE DupRank > 1
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```



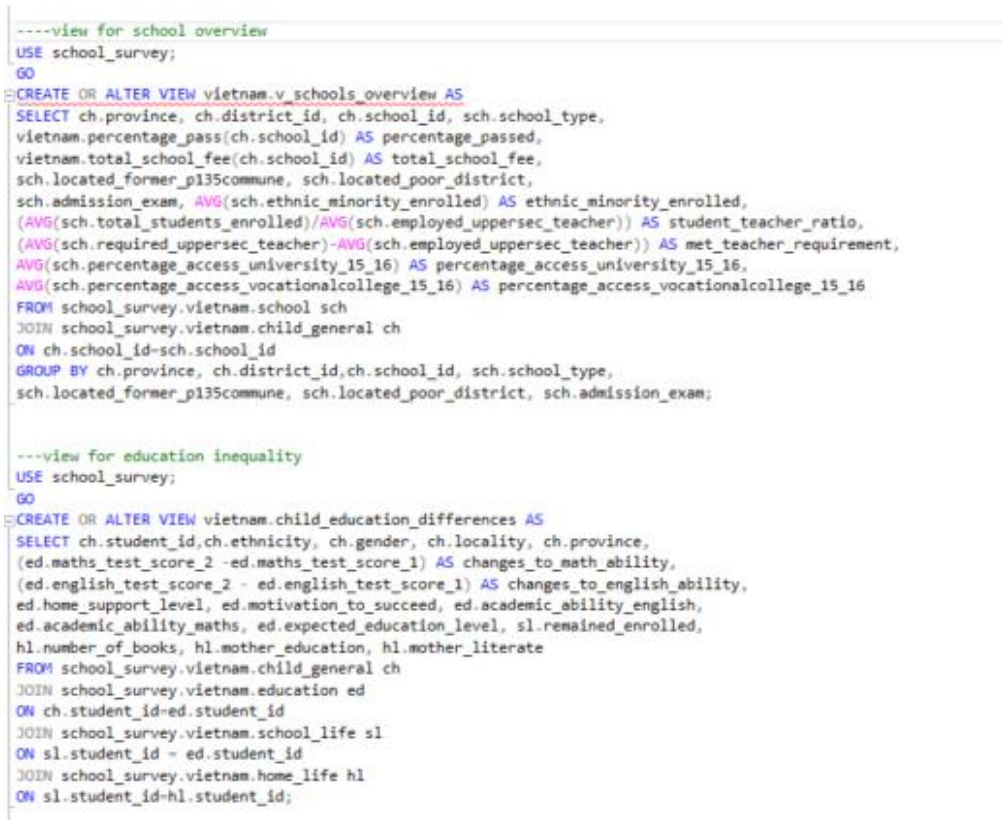
```
--trigger which prevents child general info table being deleted
USE school_survey;
GO
CREATE OR ALTER TRIGGER vietnam.not_delete_gen_info
ON school_survey.vietnam.child_general
INSTEAD OF DELETE
AS
IF EXISTS (SELECT* FROM school_survey.vietnam.child_general)
DECLARE @Error NVARCHAR(100)
SET @Error = 'Cannot delete from this table'
BEGIN
RAISERROR(@Error, 16, 1 )
ROLLBACK TRANSACTION
END
GO
```

To allow for a general overview of the schools recorded in the study and their characteristics, which could obviously be useful when trying to understand the current situation of the Vietnamese education system, a view was created which contained general information about the schools. Firstly, a view was created which produced the number of schools for each province, district, and school type (‘v_school_number’). This allows investigators to see the number of school types for each area to see if there is any difference and identify how many schools are available in each study area. Understanding this is important because children may not have any choice in the school they have to attend if there are few in the area, and if this school is of poor quality their education may suffer. However, having multiple available schools in an area creates healthy competition which forces schools to try to compete and improve their standards, especially as many schools in Vietnam are paid for by attendees to some extent. As many students have to pay for their secondary education, the function ‘total_school_fee’ was created which estimated the total costs a school may charge before considering any fee reduction. The cost of secondary education has been found to deter some poorer families from sending their children (Rolleston and Iyer, 2019). Therefore, more expensive schools may have less poorer children in attendance. Additionally, as attending secondary education is not compulsory in Vietnam (Rolleston and Iyer, 2019), those who do attend would aim to gain the qualification at the end of their studies. Therefore, a good measurement of school quality is the percentage of students who passed their end of year 12 exams out of the total year 12 student population. A function called ‘percentage pass’ was created which could calculate this using the school’s ID.



```
--number of schools in each province and district by school type
CREATE OR ALTER VIEW vietnam.v_school_number AS
SELECT ch.province, ch.district_id, sch.school_type, COUNT(DISTINCT ch.school_id) AS school_number
FROM school_survey.vietnam.school sch
JOIN school_survey.vietnam.child_general ch
ON ch.school_id=sch.school_id
GROUP BY ch.province, ch.district_id, sch.school_type;

--total cost of school for students
USE school_survey;
GO
CREATE OR ALTER FUNCTION vietnam.total_school_fee (@school_id VARCHAR(25))
RETURNS NUMERIC AS
BEGIN
    DECLARE @school_cost NUMERIC;
    SET @school_cost=(SELECT (grade10_tuitionfee+grade10_lunchfee+grade10_extraclassfee+grade10_textbookfee+grade10_uniformfee
    +grade10_schoolconstructionfee+grade10_parentsfundfee+grade10_traituyenfee+grade10_healthinsurancefee
    +grade10_schoolsecurityfee+grade10_extracurricularfee)
    FROM school_survey.vietnam.school
    WHERE school_id=@school_id);
    RETURN @school_cost
END;

--pass rate for areas
USE school_survey;
GO
CREATE OR ALTER FUNCTION vietnam.percentage_pass (@school_id VARCHAR(25))
RETURNS NUMERIC AS
BEGIN
    DECLARE @percentage_passed NUMERIC;
    DECLARE @passed INT;
    SET @passed=(SELECT AVG(passed_graduation_exam_15_16) FROM school_survey.vietnam.school WHERE school_id=@school_id
    GROUP BY school_id)
    IF (@passed = 0) SET @percentage_passed=0;
    ELSE SET @percentage_passed = (SELECT (AVG(passed_graduation_exam_15_16)/AVG(grade12_enrolled_15_16)*100)
    FROM school_survey.vietnam.school WHERE school_id=@school_id
    GROUP BY school_id);
    RETURN @percentage_passed;
END;
```

To allow for analysis of education inequality, views first needed to be created containing the required data. The first view created (‘v_schools_overview’) contained characteristics associated with schools, such as student-teacher ratio and the percentage pass rate, which can be used to determine school quality. The second view contained examples of student characteristics where education inequality has been identified, gender and ethnicity (‘child_education_differences’). To measure the potential effects of these characteristics may have on a child’s education, characteristics which can measure the child’s ability to succeed in education are included (Rolleston and Krutikova, 2014, Nguyen, 2019, Dercon and Krishnan, 2009). Examples of these are, the home support level (level of home support has been found to influence academic achievement), subject ability, whether the child dropped out, the mother’s education level, etc.



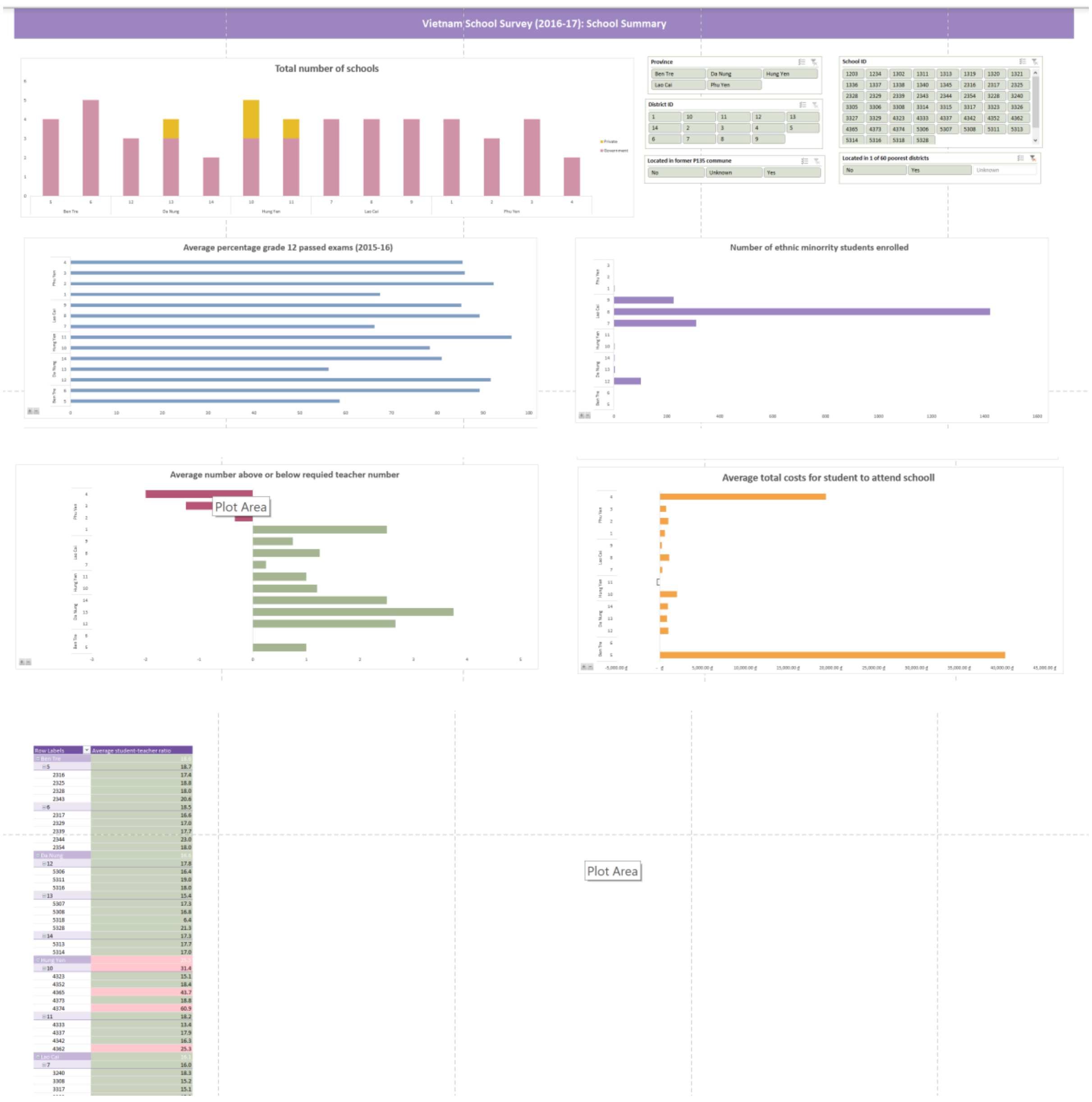
```sql
----view for school overview
USE school_survey;
GO
CREATE OR ALTER VIEW vietnam.v_schools_overview AS
SELECT ch.province, ch.district_id, ch.school_id, sch.school_type,
vietnam.percentage_pass(ch.school_id) AS percentage_passed,
vietnam.total_school_fee(ch.school_id) AS total_school_fee,
sch.located_former_p135commune, sch.located_poor_district,
sch.admission_exam, AVG(sch.ethnic_minority_enrolled) AS ethnic_minority_enrolled,
(AVG(sch.total_students_enrolled)/AVG(sch.employed_uppersec_teacher)) AS student_teacher_ratio,
(AVG(sch.required_uppersec_teacher)-AVG(sch.employed_uppersec_teacher)) AS met_teacher_requirement,
AVG(sch.percentage_access_university_15_16) AS percentage_access_university_15_16,
AVG(sch.percentage_access_vocationalcollege_15_16) AS percentage_access_vocationalcollege_15_16
FROM school_survey.vietnam.school sch
JOIN school_survey.vietnam.child_general ch
ON ch.school_id=sch.school_id
GROUP BY ch.province, ch.district_id,ch.school_id, sch.school_type,
sch.located_former_p135commune, sch.located_poor_district, sch.admission_exam;


---view for education inequality
USE school_survey;
GO
CREATE OR ALTER VIEW vietnam.child_education_differences AS
SELECT ch.student_id,ch.ethnicity, ch.gender, ch.locality, ch.province,
(ed.maths_test_score_2 -ed.maths_test_score_1) AS changes_to_math_ability,
(ed.english_test_score_2 - ed.english_test_score_1) AS changes_to_english_ability,
ed.home_support_level, ed.motivation_to_succeed, ed.academic_ability_english,
ed.academic_ability_maths, ed.expected_education_level, sl.remained_enrolled,
hl.number_of_books, hl.mother_education, hl.mother_literate
FROM school_survey.vietnam.child_general ch
JOIN school_survey.vietnam.education ed
ON ch.student_id=ed.student_id
JOIN school_survey.vietnam.school_life sl
ON sl.student_id = ed.student_id
JOIN school_survey.vietnam.home_life hl
ON sl.student_id=hl.student_id;
```

5. Report Design

In Excel it is possible to link to the SQL server database and insert the data. However, this poses a security risk, as anyone who accesses the Excel spreadsheet will have direct access to the data in the database and can therefore access secure information or alter the data within the database. Instead, an Open Database Connectivity (ODBC) is used to connect Excel to the database via a port, which maintains independence between the two. Only the data selected inserted into the Excel sheet will be available to the user and the data is only a copy which gets updated via the connection, so any changes committed in Excel do not impact the database.

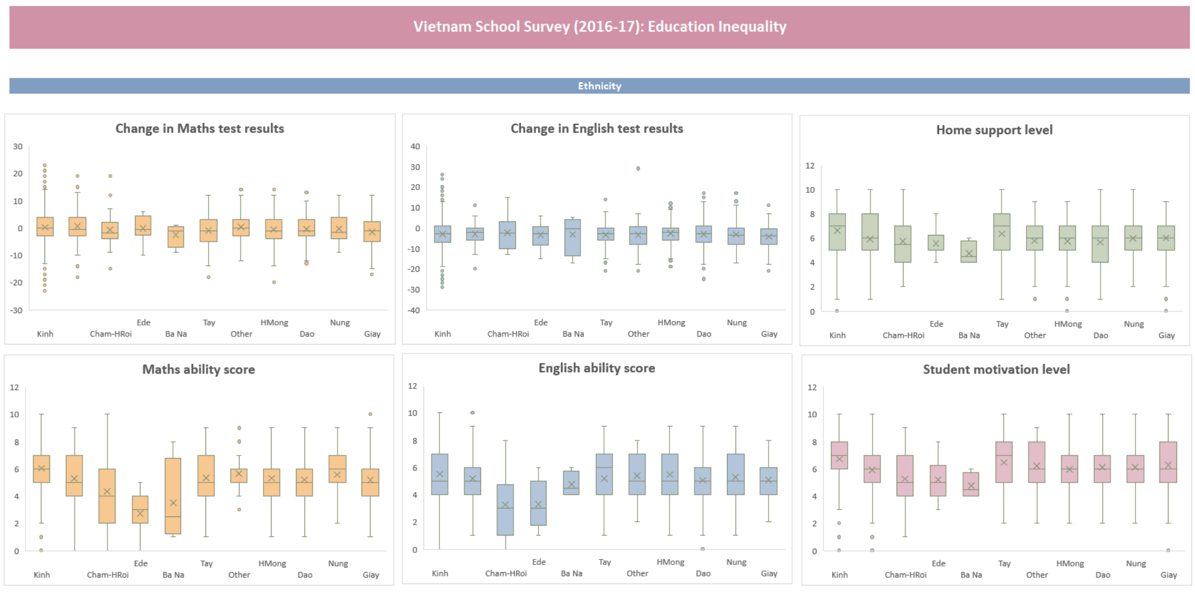
Once the school_survey database was set up as an ODCB data source, the views ‘v_schools_overview’ and ‘child_education_differences’ were inserted as tables into separate sheets in an Excel file. Users can then use these tables to filter, sort and order data according to their needs. The data in these tables can be updated to contain up-to-date data from the database via the ODCB connection. To measure education inequality in Vietnam, a dashboard for each view was produced. Data from the views was used to produce pivot tables. These could then be used to produce charts and graphs on a separate sheet altogether.

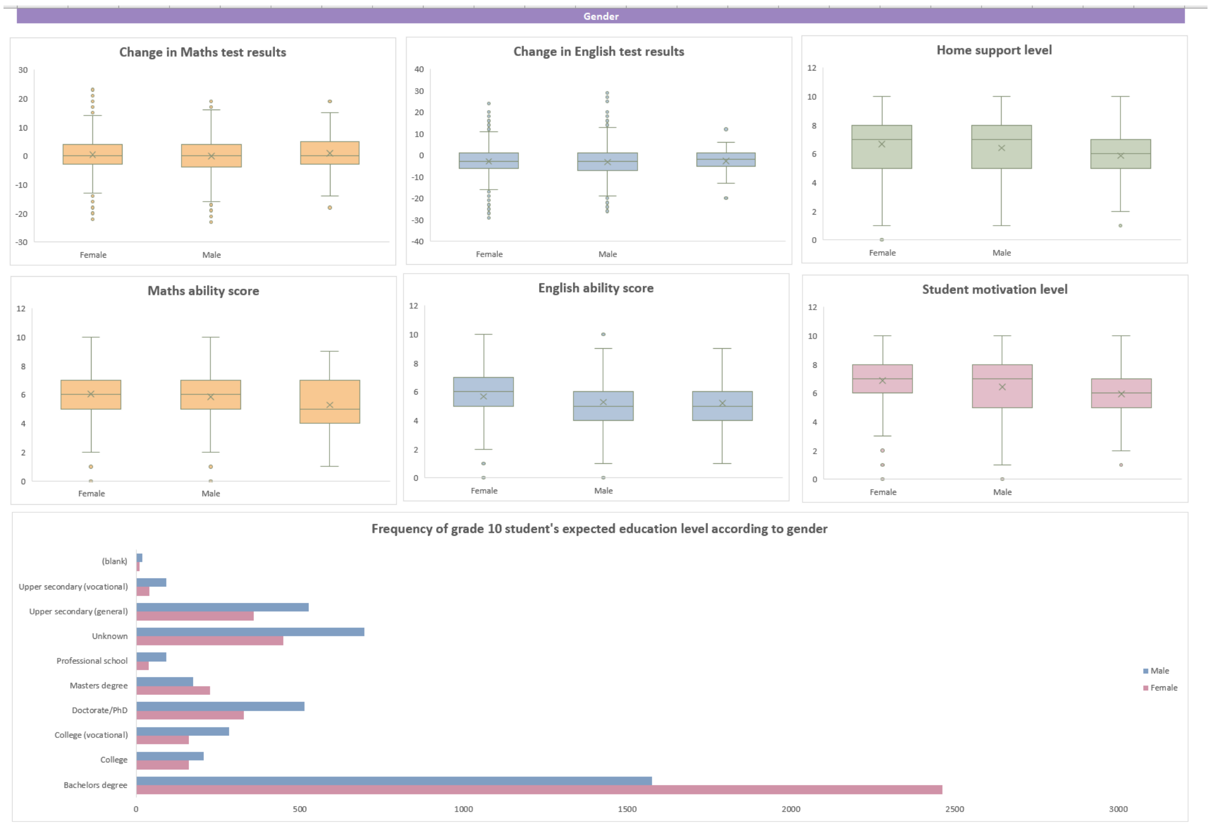
The first dashboard produced was the school summary. This contained a stacked column chart which showed the number of schools per district and province. The colour of the bar indicated the number of government schools or the number of private schools. Then a series of graphs followed which could be altered using the provided filters. Combinations of provinces, districts and schools could be selected using the filters. The graphs could then be drilled down or up to display only province level data all the way to individual school level data using the plus and minus fields on the charts. Additionally, only schools that belonged to poorer areas or were located in former P135 communes could be filtered. The first bar chart shows the average percentage of grade 12 who passed their exams in 2015-16, which can be used as a measurement of school quality. As ethnic minorities have been found to be less likely to go to secondary school and less likely to perform well (Nguyen, 2019), the number of ethnic minority students is also displayed as a bar chart. Whether the schools were above the required teacher level or below on average was indicated by the colour of the bars in the provided bar chart. Schools below the required number of teachers and so potentially of lower quality can easily be identified with the red bars. This is repeated for the student-teacher ratio table, where those with above the acceptable ratio of 23:1 (Rolleston and Krutikova, 2014) are highlighted as red. Finally, the costs of secondary education can deter some families from sending their children (Rolleston and Iyer, 2019), so the average total cost of the schools was also included as a bar chart. This dashboard is intended to provide an overview of school quality so users can potentially identify schools for further investigation.

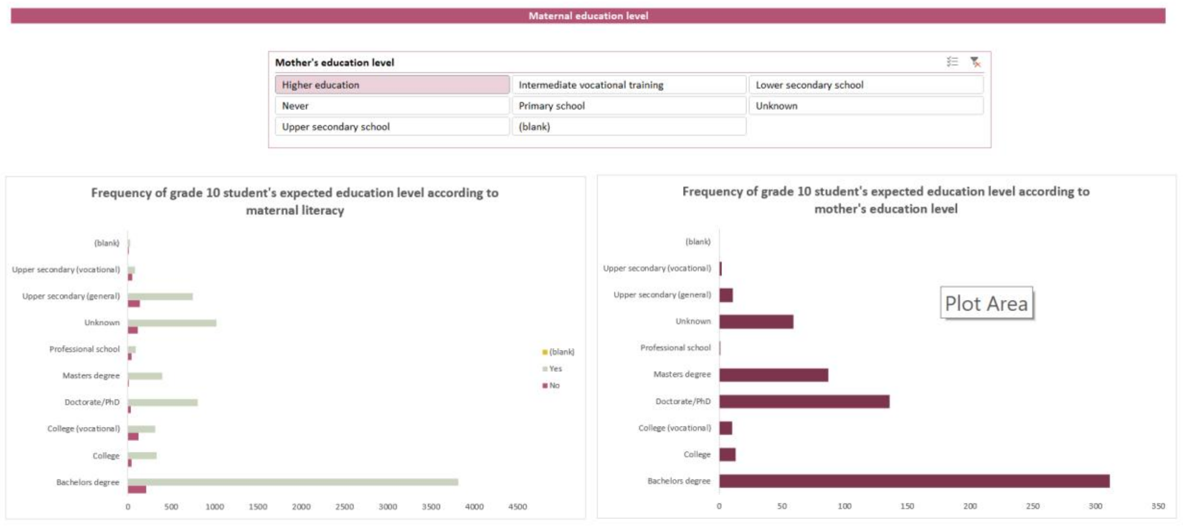

Vietnam School Survey (2016-17): School Summary
Total number of schools
Private
Government
Ben Tre
Da Nung
Hung Yen
Lao Cai
Phu Yen
Province
District ID
Located in former P135 commune
No
Unknown
Yes
School ID
Located in 1 of 60 poorest districts
Average percentage grade 12 passed exams (2015-16)
Number of ethnic minority students enrolled
Average number above or below requied teacher number
Plot Area
Average total costs for student to attend schooll

Row Labels	Average student-teacher ratio
Ben Tre	18.6
5	18.7
2316	17.4
2325	18.8
2328	18.0
2343	20.6
6	18.5
2317	16.6
2329	17.0
2339	17.7
2344	23.0
2354	18.0
Da Nung	16.6
12	17.8
5306	16.4
5311	19.0
5316	18.0
13	15.4
5307	17.3
5308	16.8
5318	6.4
5328	21.3
14	17.3
5313	17.7
5314	17.0
Hung Yen	25.5
10	31.4
4323	15.1
4352	18.4
4365	43.7
4373	18.8
4374	60.9
11	18.2
4333	13.4
4337	17.9
4342	16.3
4362	25.3
Lao Cai	16.1
7	16.0
3240	18.3
3308	15.2
3317	15.1

The second dashboard produced displayed the statistical data for measuring the effects of different student characteristics on academic ability and success. Ethnicity, gender, and maternal education have all been found to influence academic ability and success in children in Vietnam (Crookston et al., 2014, Cueto et al., 2020, Nguyen, 2019). Both the ethnicity and gender sections contain box plots to show the distribution of measurements which can be used to indicate academic ability: the difference between the Maths and English tests taken at the beginning and the end of the school year, the Maths ability score, the English ability score, the home support level, and student motivation level. Gender contains an additional bar chart which shows the frequency of expected education level according to gender, as girls have been found as more likely to obtain further study compared to boys (Cueto et al., 2020). For the maternal education section, the mother's level of education could be selected to show the frequency of students expected education level according to this selection, to see if the mother's education level corresponds with the expected education level with the highest frequency. Also, frequency of expected education level for students whose mothers can read and those whose mothers cannot.







In terms of report design, there are a number of guidelines recommended which were applied here (Shneiderman et al., 2016). Fields need to be grouped and presented in a logical manner, so the dashboards were separated into their broader topics. In the second dashboard, the charts were grouped into the characteristic which could influence academic achievement. Terms used to describe the charts and the measurements used understandable terminology. Additionally, a consistent colour scheme was applied using Excel's design tab. The use of colour allows the user to positive or negative results through using green and red, to differentiate between gender results, using red for females and blue for males and to link characteristics and measurements

6. Database security

Besides the methods of security which have been applied in the database already, there are other recommended steps which can be taken to improve the security of the database. It is possible to encrypt data to protect it from any threats or attempts to access, particularly if this data is deemed sensitive (Connolly and Begg, 2015). SQL Server offers multiple functions which can be used to encrypt data, such as ‘ENCRYPTBYKEY’ and ‘ENCRYPTBYPASSPHRASE’ (Mukherjee, 2019). Besides tools within the database, it is also recommended that external security measures are also taken (Mukherjee, 2019, Elmasri and Navathe, 2017). For example, the hardware used to run SQL server must be fault tolerant. Should one of the components fail the server can continue to run without failure. A potential option is using Redundant Array of Independent Disks (RAID) which allows data to be segmented and distributed across multiple disks (Connolly and Begg, 2015). Additional external security can involve ensuring the operating system used is up-to-date with the latest upgrades to security and that there is an operational firewall in place (Connolly and Begg, 2015). Finally, it is recommended that the hardware has physical security, such as locking the server hardware away and restricting access (Connolly and Begg, 2015).

7. Database backup

Backing up and recovering a database is another form of security. Should a failure occur where the database can no longer be used, a recent back up version can be restored (Connolly and Begg, 2015).

8. Conclusion

Using a database to store the data from the School Survey study can allow maintenance of data integrity, accuracy, and concurrency. It can also allow multiple users to use the data for future research through allowing users to create their own required combinations of attributes and calculated fields, without impacting the stored data. In the future, the organisation could store further study data, including extensions of the study used in this task, as well as other related studies conducted as part of the project.

Task 3: Crime profiler

Databases offer the opportunity to ensure an organisation's data is maintained through protecting data integrity, data concurrency and security. The storage of data in one central location can allow multiple users to access required data to meet organisation objectives and goals efficiently and quickly. This is especially important with the amount of data generated in the present age. This task aims to import crime report data from Greater Manchester and LSOA population data into a database, with the aim of enhancing researchers ability to investigate crime rates across Greater Manchester.

1. Introduction

For a long time, data was stored independently in separate files. This approach has a lot of drawbacks which can make business use inefficient (Connolly and Begg, 2015). Linking related data kept in separate files required creating new files to store combined data. Data could often be duplicated across departments and teams. Any updating of data could require updating data across multiple files, which will likely create data inconsistencies and future errors. In the end, any novel user requirements of the data would have to be custom-written and produce by the IT team, increasing the amount of time business tasks could take. With the increasing amounts of data produced, this system would only become more difficult (Connolly and Begg, 2015).

The development of databases has overcome many of these drawbacks. A database is a collection of related data which meets the needs of the organisation (Connolly and Begg, 2015). They provide a centralised location for all relevant and useful data, in the same format, allowing users to efficiently access required data. As related data is linked within the database, users are able to access combinations of data from different tables as required without the need for custom files and programs to be created (Connolly and Begg, 2015). The Database Management System is the software which allows users to interact with the stored data. They also provide database security, protect data integrity, create a concurrency control system so multiple users can access the system at one time, and a recovery system which reverts the database back to a previous consistent state in the event of failure, preventing data loss (Connolly and Begg, 2015). SQL Server is a popular DBMS software developed by Microsoft, which offers all of these services. Overall, databases allow organisations to speed up their analysis and therefore decision making process, and allow organisations to collect more data than they previously would have been able to (Connolly and Begg, 2015, Elmasri and Navathe, 2017).

This task involves using SQL server to import crime report data from the beginning of 2017 to the end of 2018 in Greater Manchester into a database and join reports with corresponding LSOA population data. This will allow the generation of an LSOA-wise summary of crime rates for different crime types in a Microsoft Excel sheet and the production of heatmaps using the software QGIS.

2. Design Rationale

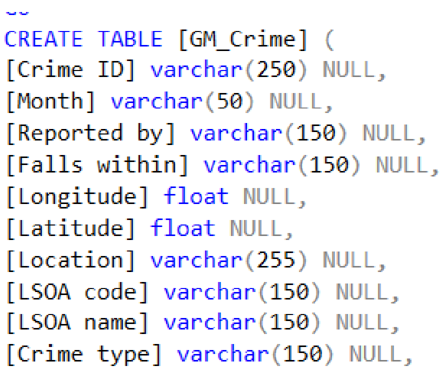
SQL Server is an example of a Relational Database Management System, which is based on the relational data model proposed by Codd (Codd, 1972). All data is represented in tables (relations) with columns (attributes) and rows (tuple) which contain one value per attribute. The relational model must meet a set of objectives: allow data independence and allow for normalisation which deals with problems relating to data semantics, consistency, and redundancy. To ensure these objectives are met, relations and attributes must have a distinct name, attribute values must be the same type, tuples must be distinct, each attribute contains a single value per tuple, and the order of tuples and attributes must not be significant (Connolly and Begg, 2015). There are a number of design considerations which help to ensure these requirements are met, including normalisation of relations and setting constraints, which will be discussed in detail later (Elmasri and Navathe, 2017).

Before database implementation, conceptual design of the database is essential in order for the database to meet the goals of the users. Understanding the overall business goals and the users requirements of the database allows the selection of appropriate data and how this data should be organised within the database (Connolly and Begg, 2015). The aim of the required database is to combine crime report data for Greater Manchester with LSOA population data to allow the generation of a report for monitoring crime rates across Greater Manchester. For the conceptual design of the database, it is popular to follow the entity-relationship (ER) model. The first step is to identify entities and their attributes. Entities are ‘objects’ with their own independent existence and can be either physical or abstract (Connolly and Begg, 2015). Each entity then has corresponding attributes which describe the properties of interest to the users (Elmasri and Navathe, 2017). In this instance, each crime report can be considered to be an entity and details recorded for each crime reported are considered to be the attributes. The attributes here can be considered to be single-valued attributes each crime report attribute has one value recorded. Any data missing data for each crime needs to be filled in with a NULL value. However, one attribute must not have any NULL values, and this is the key attribute. The primary key attribute is distinct for each entity so can be used to identify different entities (Connolly and Begg, 2015). In the crime report data, many of the crime ID values were missing and so this column could not be used as a primary key. Instead, a new column was created called ‘ID’ which could be used to uniquely identify each distinct crime report. Often in databases there are multiple entities which relate to each other and so can be linked together via these relationships. However, in this instance there is only one entity, so this type of mapping is not necessary.

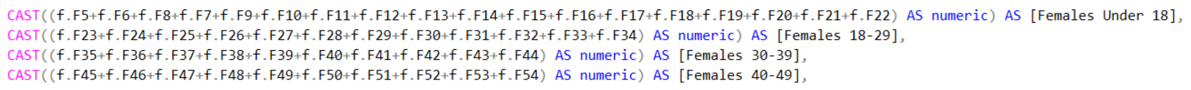
3. Design Considerations

The next stage is to consider the logical database design. This translates the conceptual model into a logical model where the model structure and ability to manage user requirements are evaluated. The stages of developing a logical data model include, normalisation, creating integrity constraints, check transactions can be handled by the model and that the model is secure (Connolly and Begg, 2015),

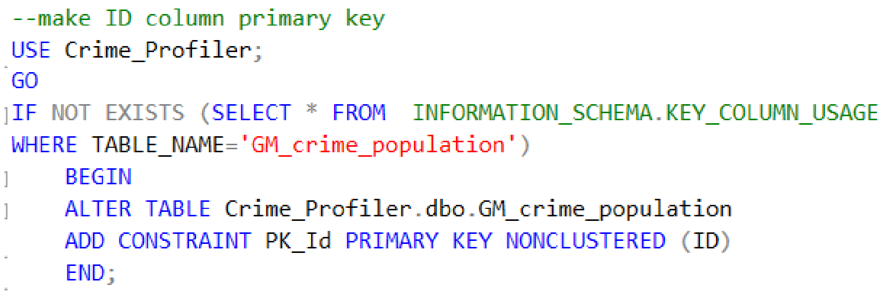
Constraints are necessary to consider in order to ensure data integrity is maintained. These are rules which prevent altering of information in the database which would impact the accuracy or quality of information stored. Constraints can be domain constraints or entity integrity constraints (Elmasri and Navathe, 2017). Domain constraints involve setting column data types. In this instance, columns such as the ‘Latitude’ and ‘Longitude’ were set as floats and the population values, like ‘Females 18-29’ were set as numeric data. This ensures that strings cannot be inputted into these columns without causing an error. Entity integrity constraints involve preventing primary key values from being null, which in this instance is the ‘ID’ column.



```
CREATE TABLE [GM_Crime] (
[Crime ID] varchar(250) NULL,
[Month] varchar(50) NULL,
[Reported by] varchar(150) NULL,
[Falls within] varchar(150) NULL,
[Longitude] float NULL,
[Latitude] float NULL,
[Location] varchar(255) NULL,
[LSOA code] varchar(150) NULL,
[LSOA name] varchar(150) NULL,
[Crime type] varchar(150) NULL,
```

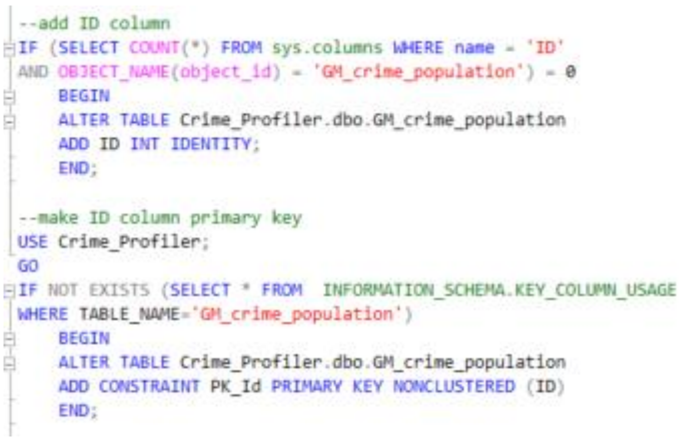


```
CAST((f.F5+f.F6+f.F8+f.F7+f.F9+f.F10+f.F11+f.F12+f.F13+f.F14+f.F15+f.F16+f.F17+f.F18+f.F19+f.F20+f.F21+f.F22) AS numeric) AS [Females Under 18],
CAST((f.F23+f.F24+f.F25+f.F26+f.F27+f.F28+f.F29+f.F30+f.F31+f.F32+f.F33+f.F34) AS numeric) AS [Females 18-29],
CAST((f.F35+f.F36+f.F37+f.F38+f.F39+f.F40+f.F41+f.F42+f.F43+f.F44) AS numeric) AS [Females 30-39],
CAST((f.F45+f.F46+f.F47+f.F48+f.F49+f.F50+f.F51+f.F52+f.F53+f.F54) AS numeric) AS [Females 40-49],
```



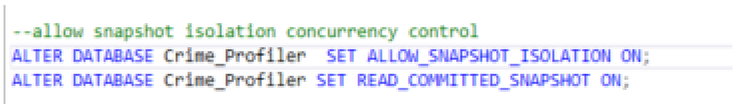
```
--make ID column primary key
USE Crime_Profiler;
GO
IF NOT EXISTS (SELECT * FROM  INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_NAME='GM_crime_population')
    BEGIN
    ALTER TABLE Crime_Profiler.dbo.GM_crime_population
    ADD CONSTRAINT PK_Id PRIMARY KEY NONCLUSTERED (ID)
    END;
```

In terms of relational databases, only the first form of normalization is essential (Connolly and Begg, 2015). First normal form requires each cell to contain a single value, each column contains the same type of entry and rows need to be unique. The CSV files containing the recorded data for each individual report already follow this form. The use of unique identifiers for each crime would ensure that each row is unique, hence the addition of the ‘ID’ column. Additionally, each column only contains a set value which was determined as part of the standard process for recording crimes. Additional normal forms are optional (Connolly and Begg, 2015). Second normal form requires all attributes which are not key columns to be dependent on the key. In this instance the primary key is the created ‘ID’ attribute. This also meets the third normal form requirements of all columns can only be determined by the table key. Additionally, running queries can sometimes produce errors which prevent them from continuing to run or produce unexpected results (Mukherjee, 2019). To raise awareness for any errors occurring, many sections of code include ‘TRY… CATCH’ statements which will print error messages should an error occur, allowing administrator intervention.



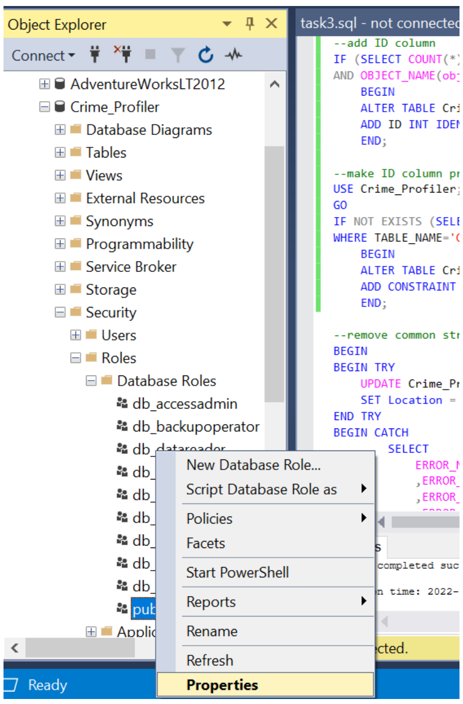
```sql
--add ID column
IF (SELECT COUNT(*) FROM sys.columns WHERE name = 'ID'
AND OBJECT_NAME(object_id) = 'GM_crime_population') = 0
    BEGIN
    ALTER TABLE Crime_Profiler.dbo.GM_crime_population
    ADD ID INT IDENTITY;
    END;

--make ID column primary key
USE Crime_Profiler;
GO
IF NOT EXISTS (SELECT * FROM  INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_NAME='GM_crime_population')
    BEGIN
    ALTER TABLE Crime_Profiler.dbo.GM_crime_population
    ADD CONSTRAINT PK_Id PRIMARY KEY NONCLUSTERED (ID)
    END;
```

In multi-user systems, many different users can access the database at the same time. This can create issues with database concurrency if two transactions (units of work) run at the same time. For example, the lost update problem is caused when two transactions are interleaved, and one transaction reads the data just before another transaction updates the data. This means the transaction will contain old data and so if this transaction also updates the data, any changes made by the previous will be lost. Another issue involves the temporary update (dirty read) problem, where an update made fails and so requires rolling back to the previous data but prior to rolling back this updated data is read by another transaction. The incorrect summary problem involves aggregating transactions obtaining a mixture of updated data and yet-to-be updated data, so is neither an accurate summary of the old data or the updated data. The unrepeatable read problem is caused when a transaction conducts two reads with another transaction updating the value in between, so the transaction reads two different values. Overall, this leads to loss of data integrity and validity. A number of algorithms have been developed to allow transaction concurrency to occur without compromising the data, each with their own strengths and limitations. The method chosen for this database is snapshot isolation, which is a form of optimistic concurrency control. This method involves taking a ‘snapshot’ of the data at the start of the transaction which transactions are conducted on rather than the actual data and once finished any updates can be applied. This method allows reads to occur without needing read ‘locks’, which only allow one read transaction to look at the data at one time, so all read transactions can occur simultaneously. Any transactions which are making any changes to the data will still require locks. This system is useful for this database as the data has already been pre-recorded for the study so should require further editing or updating. It is likely that users will just be using read transactions and any updating of data would be done via bulk insert. In SQL, this concurrency control method is activated within the database using the code in the figure below. Transactions will either be committed if the query is successful or rolled back so changes are not committed to the table.



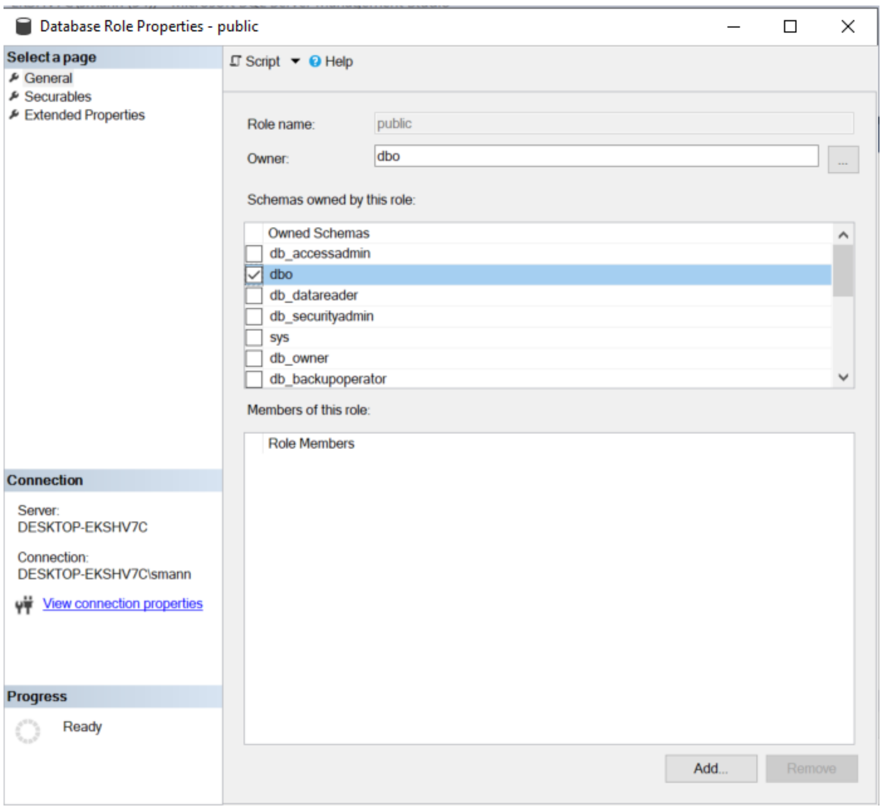
```sql
--allow snapshot isolation concurrency control
ALTER DATABASE Crime_Profiler  SET ALLOW_SNAPSHOT_ISOLATION ON;
ALTER DATABASE Crime_Profiler SET READ_COMMITTED_SNAPSHOT ON;
```

Many users are expected to be able to use the database to access data, which can put the data at risk of damage (whether intentional or unintentional) or leaking of protected information. Fortunately, in this case all crime has been anonymised regarding the people involved. There is still the risk of damage to the data held in the database which would impact the ability to use the data for its intended purpose or for data to be leaked to the outside and used without permission by others. Restricting the database access for individuals in a way which they are can only access what is required to complete their task helps to secure the database by reducing the number of individuals with complete access (Connolly and Begg, 2015). Within the database, users can be created and attached to a particular login. There are two options for users to login to the SQL database, using SQL server login or Windows authentication. It is recommended by Microsoft to use Windows authentication as this is more secure (Mukherjee, 2019). Users can then be given particular privileges which restrict them from accessing all areas of the database and restrict what they are able to do within the database, such as being able to update or change the data in any way. If there are many users, it is probably better to assign users to a role where anyone with the same role has the same privileges. All users are automatically assigned the public role, so this should be the lowest level of access a user is awarded. The public role for this database was only given access to the poverty schema and was not allowed to view/update/alter tables. Views are an excellent method of security, as they only provide individuals access to the attributes required for them to do their job, so any sensitive information in the tables can be withheld (Connolly and Begg, 2015). Therefore, views were used to select data which users would need to investigate certain aspects the crime reports, without giving them access to all the tables.



```
--add ID column
IF (SELECT COUNT(*)
AND OBJECT_NAME(ob
    BEGIN
    ALTER TABLE Cri
    ADD ID INT IDEN
    END;

--make ID column pr
USE Crime_Profiler;
GO
IF NOT EXISTS (SELE
WHERE TABLE_NAME='C
    BEGIN
    ALTER TABLE Cri
    ADD CONSTRAINT
    END;

--remove common str
BEGIN
BEGIN TRY
    UPDATE Crime_Pr
    SET Location =
END TRY
BEGIN CATCH
        SELECT
            ERROR_N
            ,ERROR_
            ,ERROR_
```

```
completed suc
n time: 2022-
```


Database Role Properties - public

Select a page
- General
- Securables
- Extended Properties

Script ▾ Help

Role name: public

Owner: dbo

Schemas owned by this role:

	Owned Schemas
☐	db_accessadmin
☑	dbo
☐	db_datareader
☐	db_securityadmin
☐	sys
☐	db_owner
☐	db_backupoperator

Members of this role:

Role Members

Connection

Server:
DESKTOP-EKSHV7C

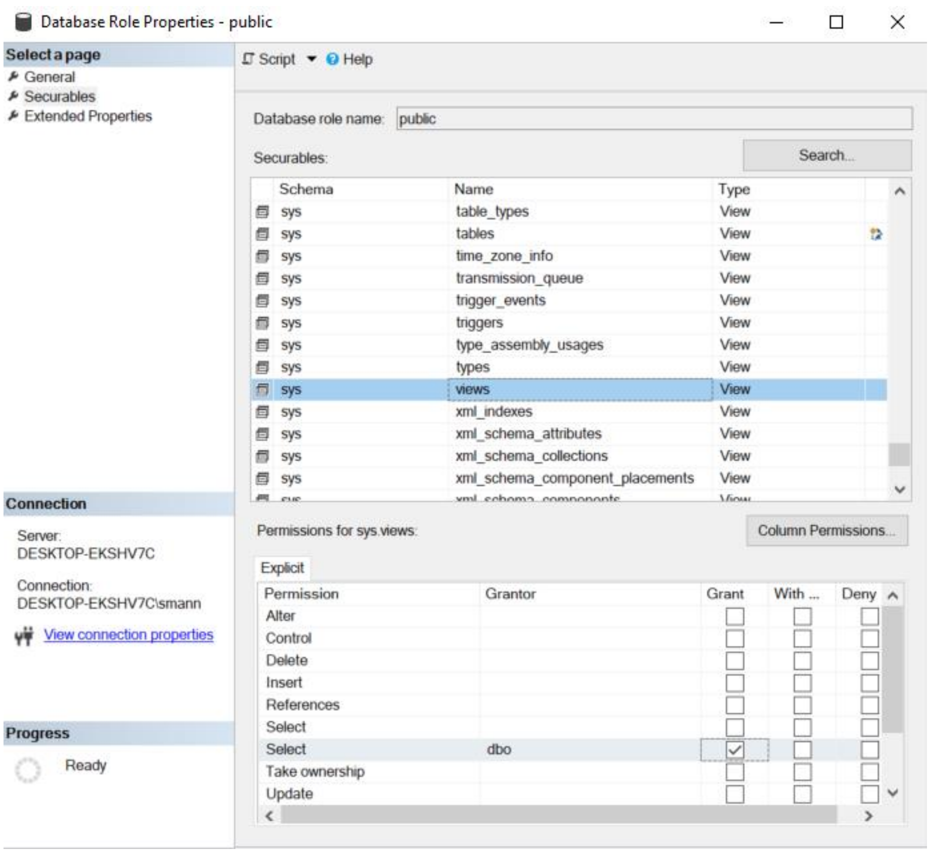
Connection:
DESKTOP-EKSHV7C\smann

View connection properties

Progress

Ready

Add... Remove


Database Role Properties - public

Select a page
- General
- Securables
- Extended Properties

Script ▾ Help

Database role name: public

Securables: Search...

Schema	Name	Type
sys	table_types	View
sys	tables	View
sys	time_zone_info	View
sys	transmission_queue	View
sys	trigger_events	View
sys	triggers	View
sys	type_assembly_usages	View
sys	types	View
sys	views	View
sys	xml_indexes	View
sys	xml_schema_attributes	View
sys	xml_schema_collections	View
sys	xml_schema_component_placements	View
sys	xml_schema_components	View

Permissions for sys.views: Column Permissions...

Explicit

Permission	Grantor	Grant	With ...	Deny
Alter		☐	☐	☐
Control		☐	☐	☐
Delete		☐	☐	☐
Insert		☐	☐	☐
References		☐	☐	☐
Select		☐	☐	☐
Select	dbo	☑	☐	☐
Take ownership		☐	☐	☐
Update		☐	☐	☐

Connection

Server:
DESKTOP-EKSHV7C

Connection:
DESKTOP-EKSHV7C\smann

View connection properties

Progress

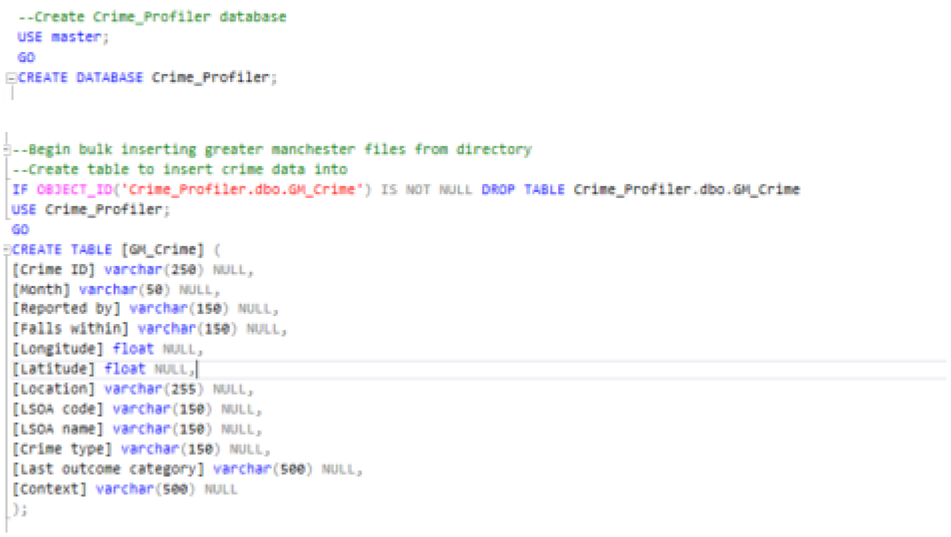
Ready

As multiple users within a database may have privileges which allow them to make edits to the database, including inserting data, creating new functions or stored procedures, etc., it is important to make comments within SQL code to explain the function of sections of code and to explain any changes made to sections of code. Additionally, it can be useful to block out sections of code which do not need to be run but may be required in a later instance.

Although not applied here, it is good convention to mark any changes to code with comments to explain the changes, the date of change and who created the change, particularly if there are multiple users with the privileges to make database changes.

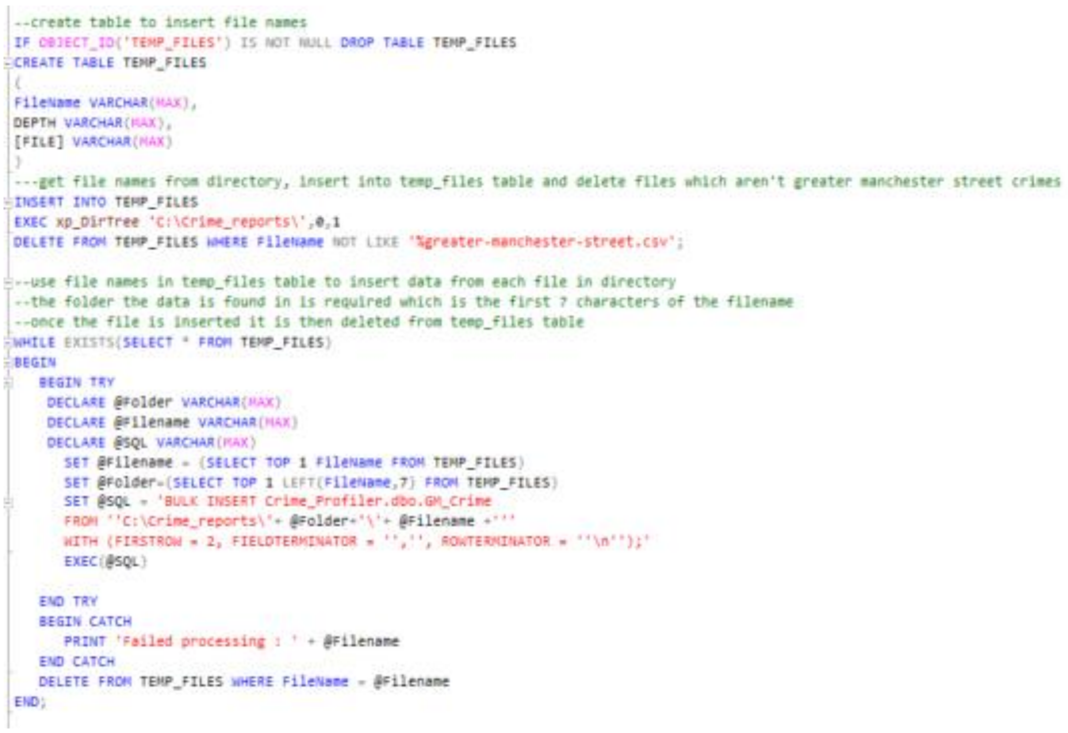
4. T-SQL statements

The database 'Crime_Profiler' was created to store the crime report data. There are numerous files available in the zip file downloaded from the UK police database (https://data.police.uk/data/). These are separated into subfolders containing files corresponding to each month between the beginning of 2017 to the end of 2018, with each file within also corresponding to crime data from specific reports and areas. The CSV files relevant for this database are those containing 'greater-manchester-street'. Inserting these files via the Import and Export Wizard would take an unnecessary amount of time and could lead to data being missed. Instead, it would be faster to bulk insert data into a table. As all the files are structured in the same way with the same columns and data types, a table corresponding to this was created named 'GM_Crime'.



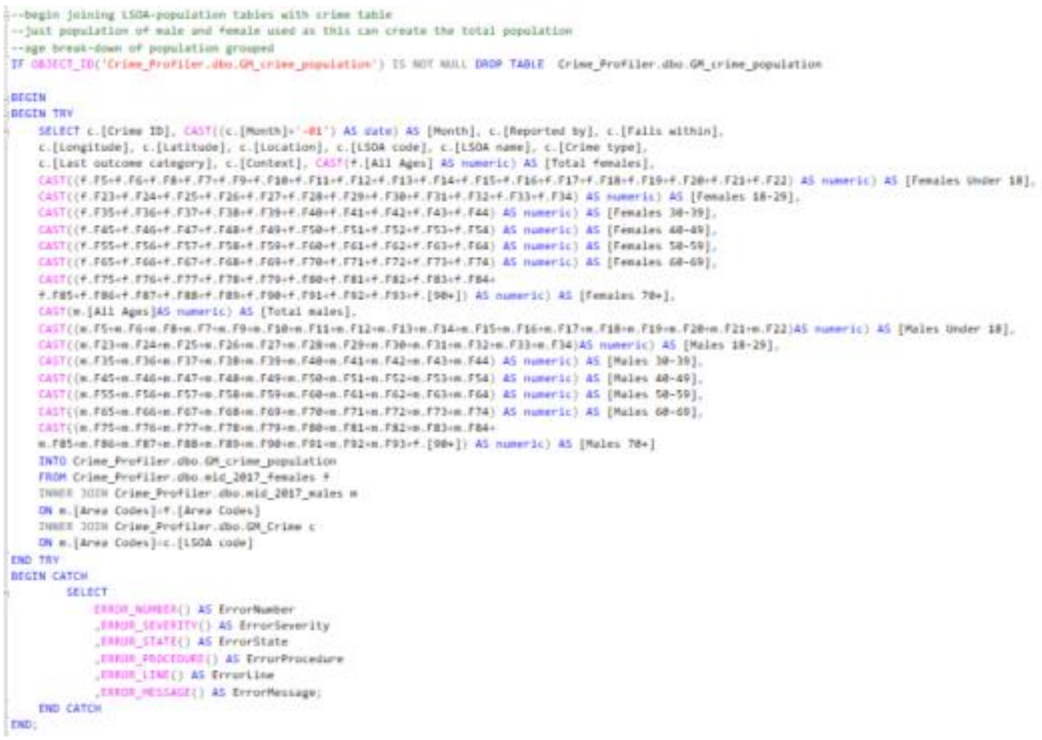
```
--Create Crime_Profiler database
USE master;
GO
CREATE DATABASE Crime_Profiler;


--Begin bulk inserting greater manchester files from directory
--Create table to insert crime data into
IF OBJECT_ID('Crime_Profiler.dbo.GM_Crime') IS NOT NULL DROP TABLE Crime_Profiler.dbo.GM_Crime
USE Crime_Profiler;
GO
CREATE TABLE [GM_Crime] (
[Crime ID] varchar(250) NULL,
[Month] varchar(50) NULL,
[Reported by] varchar(150) NULL,
[Falls within] varchar(150) NULL,
[Longitude] float NULL,
[Latitude] float NULL,
[Location] varchar(255) NULL,
[LSOA code] varchar(150) NULL,
[LSOA name] varchar(150) NULL,
[Crime type] varchar(150) NULL,
[Last outcome category] varchar(500) NULL,
[Context] varchar(500) NULL
);
```

To begin inserting the data, a table named 'TEMP_FILES' was created which had the data from the system procedure 'xp_DirTree' inserted into it, such as file names contained within the folder of crime report data. To ensure only the relevant files are kept, any filenames which did not contain ''%greater-manchester-street.csv' were deleted from the table. Then a loop was used to bulk insert each file with the filename contained in the 'TEMP_FILE' table, into the 'GM_Crime' table. This will insert the data from all relevant files into the table. If data is recorded in CSV files and then uploaded into the folder each month, it is possible to make this whole process into a stored procedure which can run monthly to update the 'GM_Crime' table with up-to-date data. On the other hand, this database could be used to manually insert reported crime data in real time through the creation of a stored procedure which allows officers to input crime report data directly into the database.

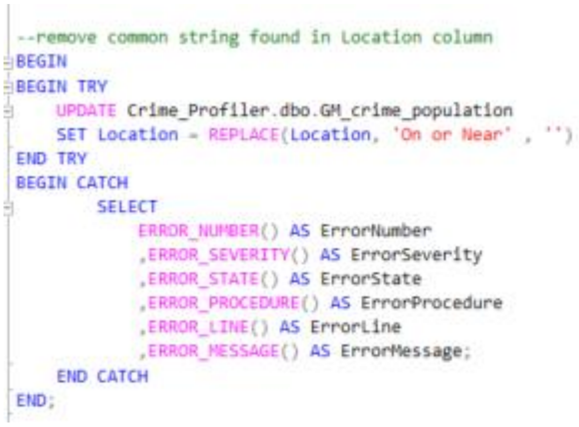


```sql
--create table to insert file names
IF OBJECT_ID('TEMP_FILES') IS NOT NULL DROP TABLE TEMP_FILES
CREATE TABLE TEMP_FILES
(
FileName VARCHAR(MAX),
DEPTH VARCHAR(MAX),
[FILE] VARCHAR(MAX)
)
---get file names from directory, insert into temp_files table and delete files which aren't greater manchester street crimes
INSERT INTO TEMP_FILES
EXEC xp_DirTree 'C:\Crime_reports\',0,1
DELETE FROM TEMP_FILES WHERE FileName NOT LIKE '%greater-manchester-street.csv';

--use file names in temp_files table to insert data from each file in directory
--the folder the data is found in is required which is the first 7 characters of the filename
--once the file is inserted it is then deleted from temp_files table
WHILE EXISTS(SELECT * FROM TEMP_FILES)
BEGIN
   BEGIN TRY
    DECLARE @Folder VARCHAR(MAX)
    DECLARE @Filename VARCHAR(MAX)
    DECLARE @SQL VARCHAR(MAX)
      SET @Filename = (SELECT TOP 1 FileName FROM TEMP_FILES)
      SET @Folder=(SELECT TOP 1 LEFT(FileName,7) FROM TEMP_FILES)
      SET @SQL = 'BULK INSERT Crime_Profiler.dbo.GM_Crime
      FROM ''C:\Crime_reports\'+ @Folder+'\'+ @Filename +'''
      WITH (FIRSTROW = 2, FIELDTERMINATOR = '','', ROWTERMINATOR = ''\n'');'
      EXEC(@SQL)

   END TRY
   BEGIN CATCH
      PRINT 'Failed processing : ' + @Filename
   END CATCH
   DELETE FROM TEMP_FILES WHERE FileName = @Filename
END;
```

The population data relating to LSOAs was only contained in one excel file and so SSMS's Import and Export Wizard, which generates a table automatically, was used to insert the data into tables. There were multiple sheets within the excel file containing total population data, male population data and female population data broken down by age. Age and gender have been found to affect the likelihood of a person committing a crime (Friedman and Rosenbaum, 1988), therefore these should be included in the database to allow further investigation using these factors. As the male and female data population could be used to calculate the total population, only these two were inserted into tables ('mid_2017_females' and 'mid_2017_males'). These two tables were then joined based on 'Area codes', with each age group combined into larger groups to reduce the number of columns. This was also joined to the table 'GM_Crime' based on the Area code being the same as the LSOA code in the 'GM_Crime' table. This created a table called 'GM_crime_population'. The 'Location' column contained the string 'On or Near' before every location recorded. This was removed so users could conduct analysis by grouping locations if required.

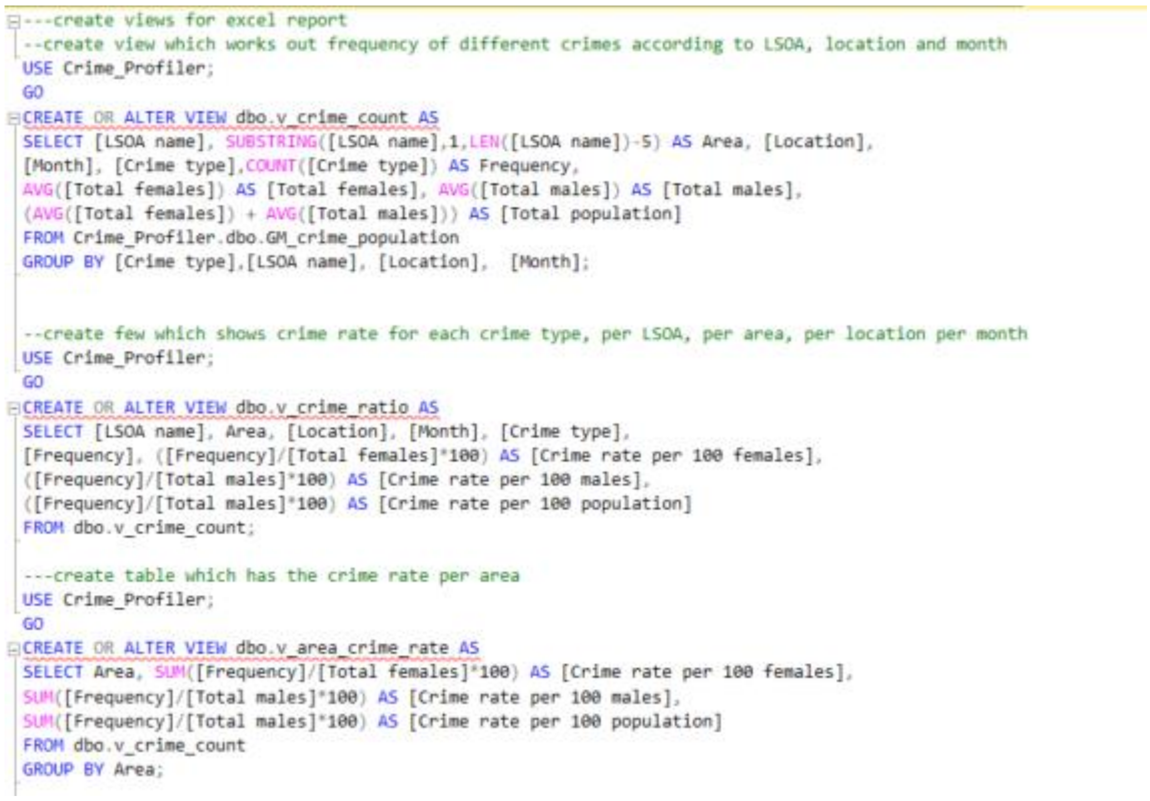


```sql
--begin joining LSOA-population tables with crime table
--just population of male and female used as this can create the total population
--age break-down of population grouped
IF OBJECT_ID('Crime_Profiler.dbo.GM_crime_population') IS NOT NULL DROP TABLE  Crime_Profiler.dbo.GM_crime_population

BEGIN
BEGIN TRY
    SELECT c.[Crime ID], CAST((c.[Month]+'-01') AS date) AS [Month], c.[Reported by], c.[Falls within],
    c.[Longitude], c.[Latitude], c.[Location], c.[LSOA code], c.[LSOA name], c.[Crime type],
    c.[Last outcome category], c.[Context], CAST(f.[All Ages] AS numeric) AS [Total females],
    CAST((f.F5+f.F6+f.F8+f.F7+f.F9+f.F10+f.F11+f.F12+f.F13+f.F14+f.F15+f.F16+f.F17+f.F18+f.F19+f.F20+f.F21+f.F22) AS numeric) AS [Females Under 18],
    CAST((f.F23+f.F24+f.F25+f.F26+f.F27+f.F28+f.F29+f.F30+f.F31+f.F32+f.F33+f.F34) AS numeric) AS [Females 18-29],
    CAST((f.F35+f.F36+f.F37+f.F38+f.F39+f.F40+f.F41+f.F42+f.F43+f.F44) AS numeric) AS [Females 30-39],
    CAST((f.F45+f.F46+f.F47+f.F48+f.F49+f.F50+f.F51+f.F52+f.F53+f.F54) AS numeric) AS [Females 40-49],
    CAST((f.F55+f.F56+f.F57+f.F58+f.F59+f.F60+f.F61+f.F62+f.F63+f.F64) AS numeric) AS [Females 50-59],
    CAST((f.F65+f.F66+f.F67+f.F68+f.F69+f.F70+f.F71+f.F72+f.F73+f.F74) AS numeric) AS [Females 60-69],
    CAST((f.F75+f.F76+f.F77+f.F78+f.F79+f.F80+f.F81+f.F82+f.F83+f.F84+
    f.F85+f.F86+f.F87+f.F88+f.F89+f.F90+f.F91+f.F92+f.F93+f.[90+]) AS numeric) AS [Females 70+],
    CAST(m.[All Ages]AS numeric) AS [Total males],
    CAST((m.F5+m.F6+m.F8+m.F7+m.F9+m.F10+m.F11+m.F12+m.F13+m.F14+m.F15+m.F16+m.F17+m.F18+m.F19+m.F20+m.F21+m.F22)AS numeric) AS [Males Under 18],
    CAST((m.F23+m.F24+m.F25+m.F26+m.F27+m.F28+m.F29+m.F30+m.F31+m.F32+m.F33+m.F34)AS numeric) AS [Males 18-29],
    CAST((m.F35+m.F36+m.F37+m.F38+m.F39+m.F40+m.F41+m.F42+m.F43+m.F44) AS numeric) AS [Males 30-39],
    CAST((m.F45+m.F46+m.F47+m.F48+m.F49+m.F50+m.F51+m.F52+m.F53+m.F54) AS numeric) AS [Males 40-49],
    CAST((m.F55+m.F56+m.F57+m.F58+m.F59+m.F60+m.F61+m.F62+m.F63+m.F64) AS numeric) AS [Males 50-59],
    CAST((m.F65+m.F66+m.F67+m.F68+m.F69+m.F70+m.F71+m.F72+m.F73+m.F74) AS numeric) AS [Males 60-69],
    CAST((m.F75+m.F76+m.F77+m.F78+m.F79+m.F80+m.F81+m.F82+m.F83+m.F84+
    m.F85+m.F86+m.F87+m.F88+m.F89+m.F90+m.F91+m.F92+m.F93+f.[90+]) AS numeric) AS [Males 70+]
    INTO Crime_Profiler.dbo.GM_crime_population
    FROM Crime_Profiler.dbo.mid_2017_females f
    INNER JOIN Crime_Profiler.dbo.mid_2017_males m
    ON m.[Area Codes]=f.[Area Codes]
    INNER JOIN Crime_Profiler.dbo.GM_Crime c
    ON m.[Area Codes]=c.[LSOA code]
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```



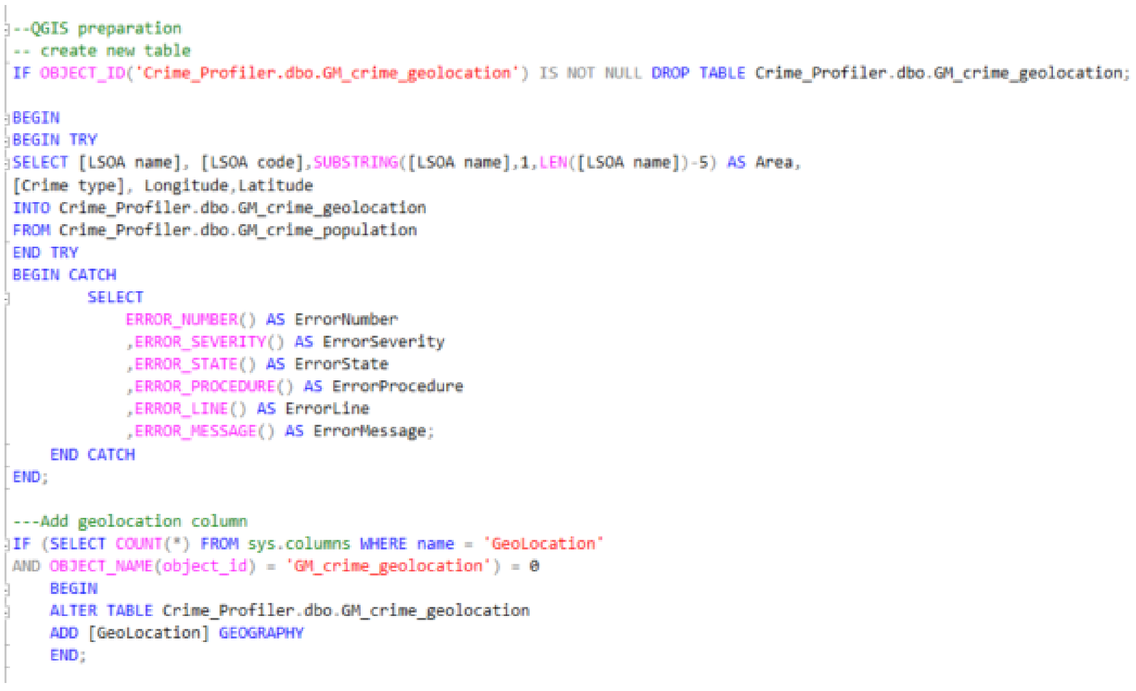
```sql
--remove common string found in Location column
BEGIN
BEGIN TRY
    UPDATE Crime_Profiler.dbo.GM_crime_population
    SET Location = REPLACE(Location, 'On or Near' , '')
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
```

To produce the LSOA-wise crime summary report, a view first needed to be created containing the required data. A first view was created called ‘v_crime_count’ which calculated the frequency of each crime type per LSOA, per location, per month. This view was then used to create another view called ‘v_crime_ratio’ which contained the frequency and crime rate per 100 people for each crime type. Crime rate is a more informative measurement than just using population alone, as each area has different population sizes. Crime rate allows for comparison between areas. An additional column called ‘Area’ was created which contained the city name extracted from the LSOA name. An additional view was created which summarised the crime rate per area called ‘v_area_crime_rate’.

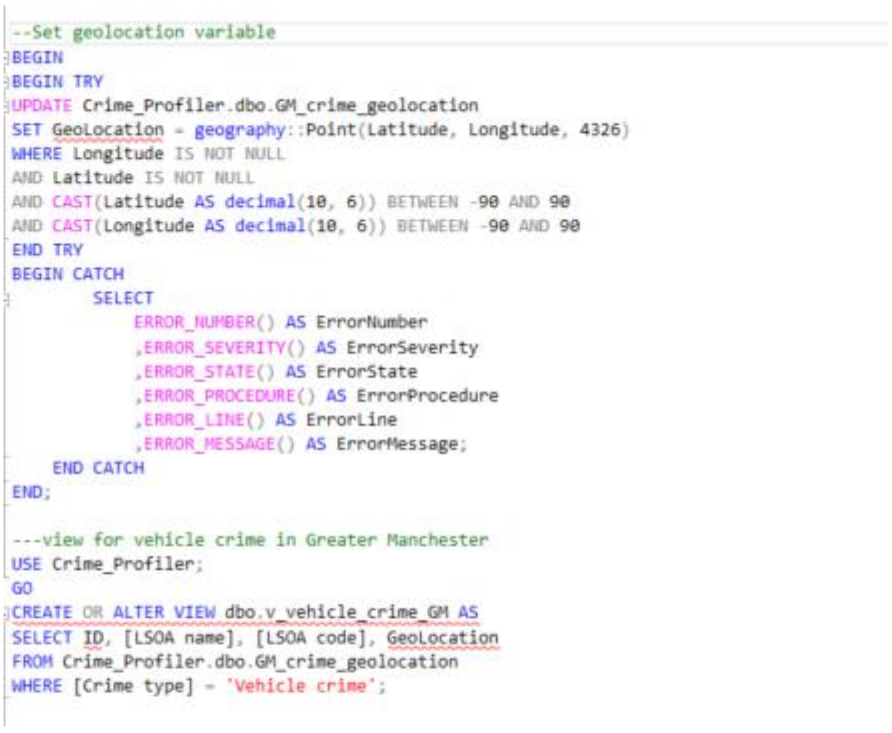


```
---create views for excel report
--create view which works out frequency of different crimes according to LSOA, location and month
USE Crime_Profiler;
GO
CREATE OR ALTER VIEW dbo.v_crime_count AS
SELECT [LSOA name], SUBSTRING([LSOA name],1,LEN([LSOA name])-5) AS Area, [Location],
[Month], [Crime type],COUNT([Crime type]) AS Frequency,
AVG([Total females]) AS [Total females], AVG([Total males]) AS [Total males],
(AVG([Total females]) + AVG([Total males])) AS [Total population]
FROM Crime_Profiler.dbo.GM_crime_population
GROUP BY [Crime type],[LSOA name], [Location],  [Month];


--create few which shows crime rate for each crime type, per LSOA, per area, per location per month
USE Crime_Profiler;
GO
CREATE OR ALTER VIEW dbo.v_crime_ratio AS
SELECT [LSOA name], Area, [Location], [Month], [Crime type],
[Frequency], ([Frequency]/[Total females]*100) AS [Crime rate per 100 females],
([Frequency]/[Total males]*100) AS [Crime rate per 100 males],
([Frequency]/[Total males]*100) AS [Crime rate per 100 population]
FROM dbo.v_crime_count;

---create table which has the crime rate per area
USE Crime_Profiler;
GO
CREATE OR ALTER VIEW dbo.v_area_crime_rate AS
SELECT Area, SUM([Frequency]/[Total females]*100) AS [Crime rate per 100 females],
SUM([Frequency]/[Total males]*100) AS [Crime rate per 100 males],
SUM([Frequency]/[Total males]*100) AS [Crime rate per 100 population]
FROM dbo.v_crime_count
GROUP BY Area;
```

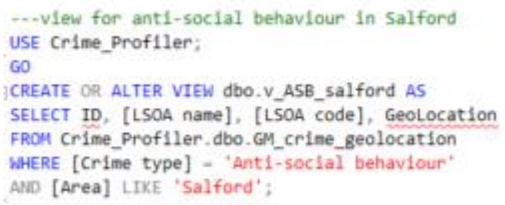
The software QGIS allows visualisation of spatial data so can be used to visualise areas which contain high levels of particular crimes. As QGIS requires geolocation data to plot data points on a map, a new table was created ('GM_crime_geolocation') which added a 'GeoLocation' column. This used the longitude and latitude of each crime reported to create a geography datatype. As an example of the capability of QGIS, two views were created. The first contained only vehicle crime across Greater Manchester. The second contained anti-social behaviour reports in Salford.



```
--QGIS preparation
-- create new table
IF OBJECT_ID('Crime_Profiler.dbo.GM_crime_geolocation') IS NOT NULL DROP TABLE Crime_Profiler.dbo.GM_crime_geolocation;

BEGIN
BEGIN TRY
SELECT [LSOA name], [LSOA code],SUBSTRING([LSOA name],1,LEN([LSOA name])-5) AS Area,
[Crime type], Longitude,Latitude
INTO Crime_Profiler.dbo.GM_crime_geolocation
FROM Crime_Profiler.dbo.GM_crime_population
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;

---Add geolocation column
IF (SELECT COUNT(*) FROM sys.columns WHERE name = 'GeoLocation'
AND OBJECT_NAME(object_id) = 'GM_crime_geolocation') = 0
    BEGIN
    ALTER TABLE Crime_Profiler.dbo.GM_crime_geolocation
    ADD [GeoLocation] GEOGRAPHY
    END;
```

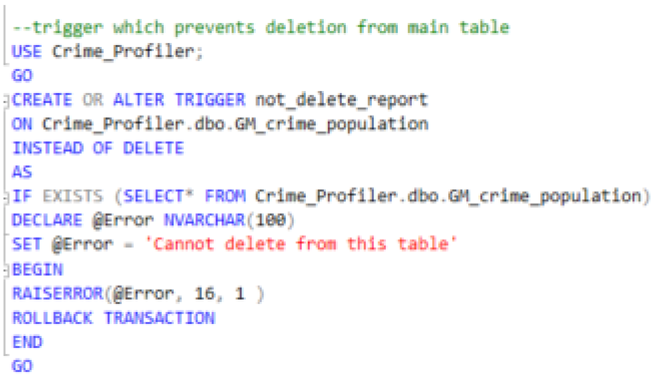


```
--Set geolocation variable
BEGIN
BEGIN TRY
UPDATE Crime_Profiler.dbo.GM_crime_geolocation
SET GeoLocation = geography::Point(Latitude, Longitude, 4326)
WHERE Longitude IS NOT NULL
AND Latitude IS NOT NULL
AND CAST(Latitude AS decimal(10, 6)) BETWEEN -90 AND 90
AND CAST(Longitude AS decimal(10, 6)) BETWEEN -90 AND 90
END TRY
BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber
            ,ERROR_SEVERITY() AS ErrorSeverity
            ,ERROR_STATE() AS ErrorState
            ,ERROR_PROCEDURE() AS ErrorProcedure
            ,ERROR_LINE() AS ErrorLine
            ,ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;

---view for vehicle crime in Greater Manchester
USE Crime_Profiler;
GO
CREATE OR ALTER VIEW dbo.v_vehicle_crime_GM AS
SELECT ID, [LSOA name], [LSOA code], GeoLocation
FROM Crime_Profiler.dbo.GM_crime_geolocation
WHERE [Crime type] = 'Vehicle crime';
```



```
---view for anti-social behaviour in Salford
USE Crime_Profiler;
GO
CREATE OR ALTER VIEW dbo.v_ASB_salford AS
SELECT ID, [LSOA name], [LSOA code], GeoLocation
FROM Crime_Profiler.dbo.GM_crime_geolocation
WHERE [Crime type] = 'Anti-social behaviour'
AND [Area] LIKE 'Salford';
```

To ensure data essential data was not lost from the main table ('GM_Crime_population'), a trigger was created which stops any deletion from occurring on this table and prints an error message. This ensures data is not lost in dependent tables and views, as well as protects data accuracy and integrity by preventing crime reports from being removed.



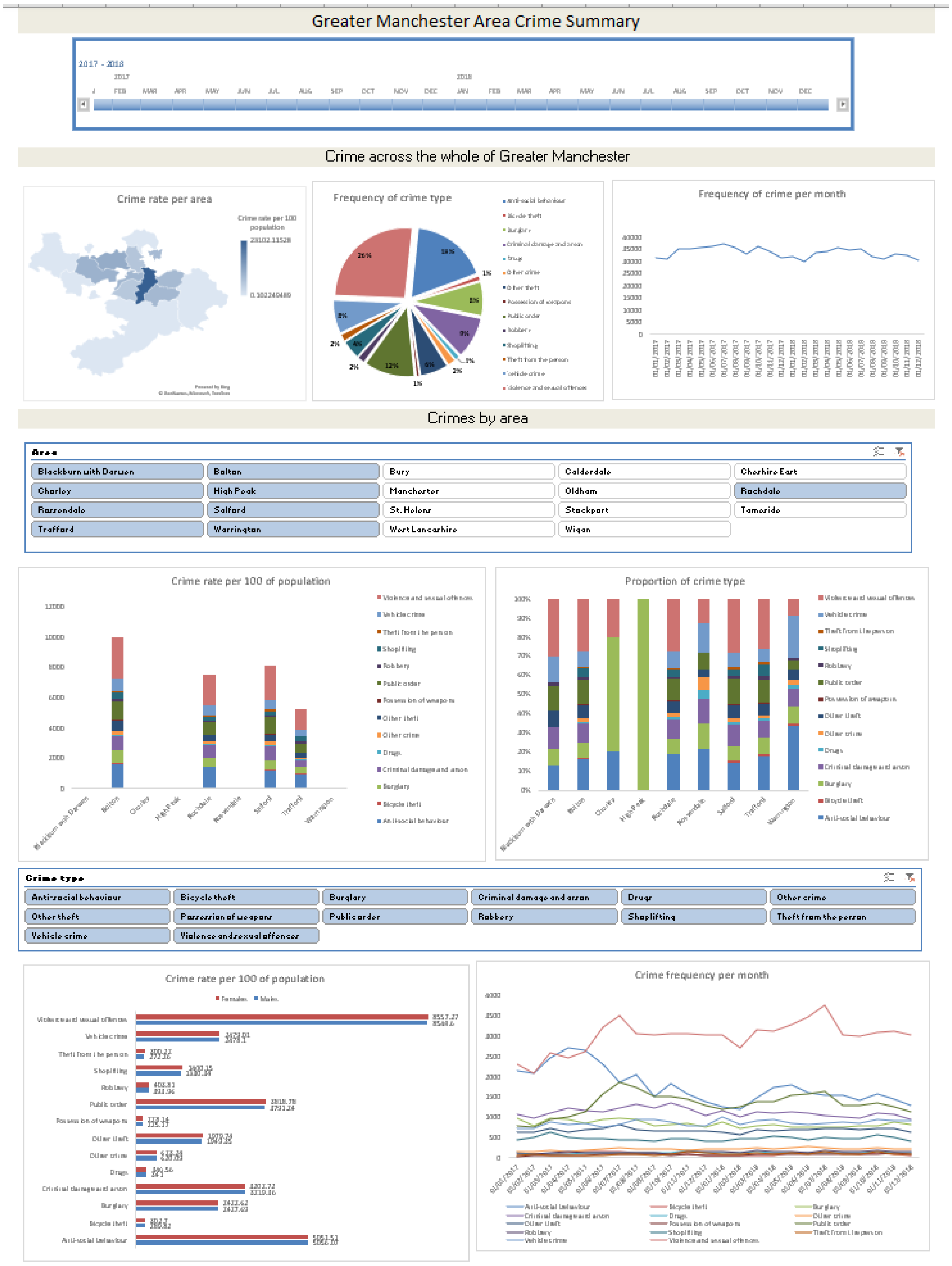
```
--trigger which prevents deletion from main table
USE Crime_Profiler;
GO
CREATE OR ALTER TRIGGER not_delete_report
ON Crime_Profiler.dbo.GM_crime_population
INSTEAD OF DELETE
AS
IF EXISTS (SELECT* FROM Crime_Profiler.dbo.GM_crime_population)
DECLARE @Error NVARCHAR(100)
SET @Error = 'Cannot delete from this table'
BEGIN
RAISERROR(@Error, 16, 1 )
ROLLBACK TRANSACTION
END
GO
```

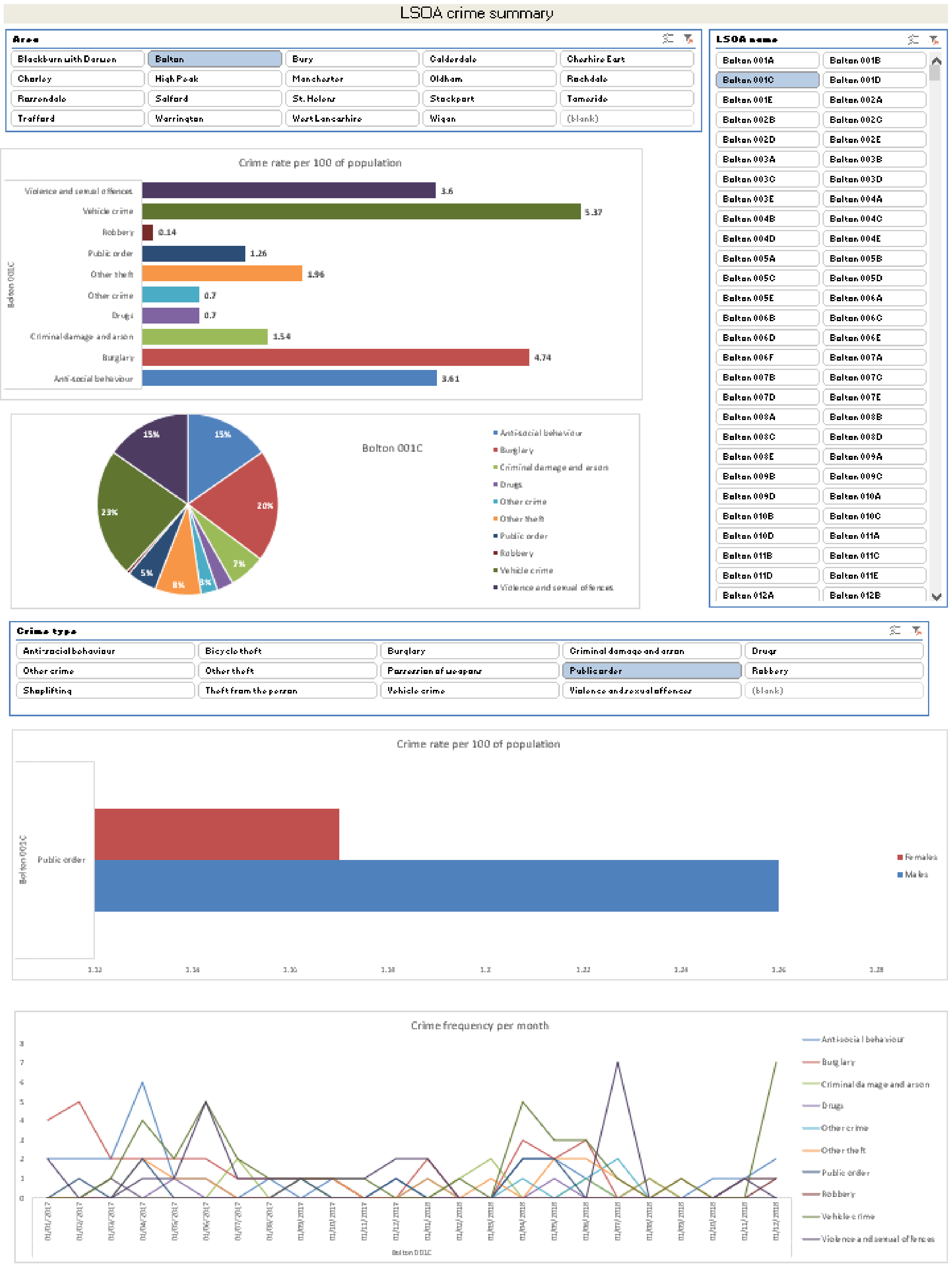
5. Report Design

In Excel it is possible to link to the SQL server database and insert the data. However, this poses a security risk, as anyone who accesses the Excel spreadsheet will have direct access to the data in the database and can therefore access secure information or alter the data within the database. Instead, an Open Database Connectivity (ODBC) is used to connect Excel to the database via a port, which maintains independence between the two. Only the data selected inserted into the Excel sheet will be available to the user and the data is only a copy which gets updated via the connection, so any changes committed in Excel do not impact the database.

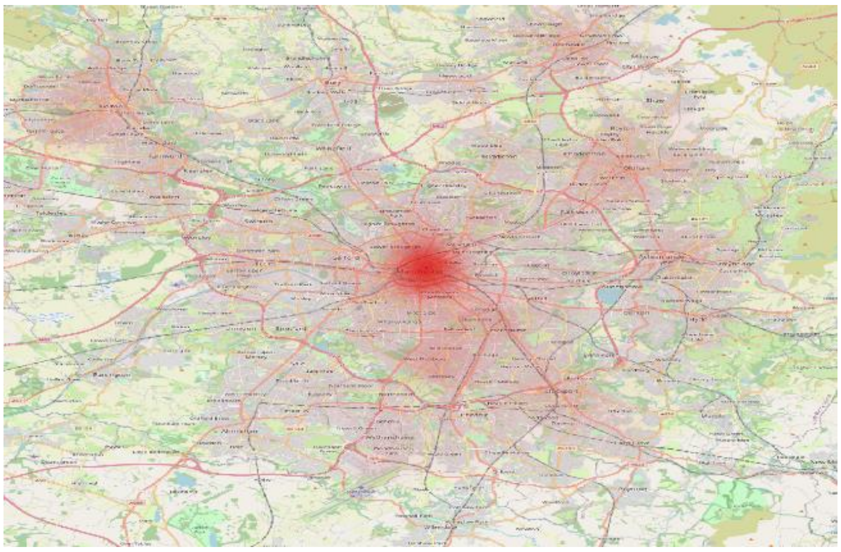
Once the Crime_Profiler database was set up as an ODCB data source, the views ‘v_crime_ratio’ and ‘v_area_crime_rate’ were inserted as tables into separate sheets in an Excel file. Users can then use these tables to filter, sort and order data according to their needs. The data in these tables can be updated to contain up-to-date data from the database via the ODCB connection. For an overall LSOA-wise crime report, a dashboard was produced. Data from the views was used to produce pivot tables. These could then be used to produce charts and graphs on a separate sheet altogether. The dashboard was split into three sections: Crime across the whole of Greater Manchester, crimes by area and LSOA crime summary. For crimes across Greater Manchester, three charts were produced; a map which coloured areas according to their population rate, a pie chart which showed the proportion of each crime type and a time-series chart which showed the total frequency of crime per month. For crimes by area the crime rate for each area was displayed in a column chart to allow direct comparison between areas and the proportion of each crime type for selected areas. Additional graphs allowed the selection of particular crime types to compare the crime rate per 100 people for the population of each gender, to see if particular crimes occur where there is a gender skew, and a time series plot which shows the frequency of a crime type per month for selected areas. Multiple areas can be selected to allow comparison for this overall section. The LSOA crime summary allows the selection of areas and subsequent individual LSOAs, with the crime rate per 100 people for each crime type presented in a bar chart, the proportion of crime types committed in the selected LSOA area in a pie chart, a bar chart containing the crime rate per 100 people for each crime type split by gender population and a times-series graph showing the frequency of each crime type reported per month for those selected. All figures in the dashboard can also be filtered by date using a slider to allow users to investigate crime reports made between certain dates and to allow the crime reports to continue to be updated over time with more recent data.

In terms of report design, there are a number of guidelines recommended which were applied here. Fields need to be grouped and presented in a logical manner, so the dashboard charts were grouped into the three topics they addressed, and the design started from the broader picture of crime across Greater Manchester to the narrower image of crime in particular LSOAs. Terms used to describe the charts and the measurements used understandable terminology. Additionally, a consistent colour scheme was applied using Excel’s design tab. The use of colour allows the user to distinguish different crime types, identify areas with higher rates of crime through colour scale and to differentiate between gender results, using red for females and blue for males.

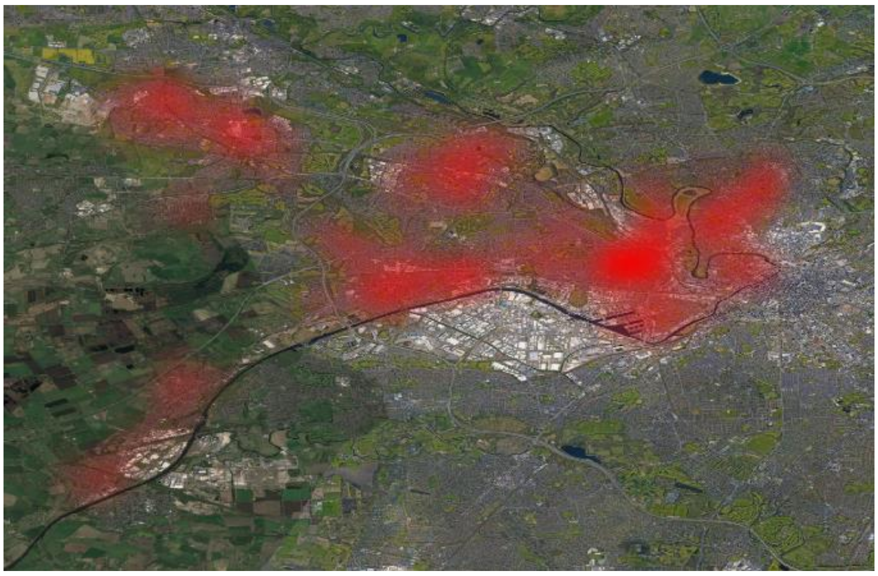

Greater Manchester Area Crime Summary
2017 - 2018
2017
2018
J
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
JAN
Crime across the whole of Greater Manchester
Crime rate per area
Crime rate per 100 population
23102.11528
0.102240489
Frequency of crime type
Anti-social behaviour
Bicycle theft
Burglary
Criminal damage and arson
Drugs
Other crime
Other theft
Possession of weapons
Public order
Robbery
Shoplifting
Theft from the person
Vehicle crime
Violence and sexual offences
Frequency of crime per month
Crimes by area
Area
Blackburn with Darwen
Bolton
Bury
Calderdale
Cheshire East
Chorley
High Peak
Manchester
Oldham
Rochdale
Rossendale
Salford
St. Helens
Stockport
Tameside
Trafford
Warrington
West Lancashire
Wigan
Crime rate per 100 of population
Proportion of crime type
Crime type
Anti-social behaviour
Bicycle theft
Burglary
Criminal damage and arson
Drugs
Other crime
Other theft
Possession of weapons
Public order
Robbery
Shoplifting
Theft from the person
Vehicle crime
Violence and sexual offences
Crime rate per 100 of population
Female
Male
8557.27
8540.6
2472.01
2470.1
300.27
272.26
1407.15
1380.34
403.81
411.96
3818.78
3793.24
179.14
175.17
1979.74
1949.26
633.34
630.03
340.56
341
2303.72
2310.16
2431.62
2437.03
304.7
289.82
5083.51
5086.20
Crime frequency per month



In QGIS, layers can be added from SQL server databases from tables/views which contain geolocation data. When mapping the instances of vehicle crime reported from the beginning of 2017 to the end of 2018 in Greater Manchester, the visual map layer used was the Open Street View map. The number of points plotted individually was much to large to try to identify any patterns of areas which had higher frequencies. Therefore a heatmap was used instead which had a colour scale of transparent to opaque red according to the number of instances occuring in particular areas. This map clearly shows that Manchester city centre had the highest amount of vehicle crime reported.



For the mapping of instances of anti-social behaviour reported from the beginning of 2017 to the end of 2018 in Salford, the visual map layer used was Google satellite view. Once again there were too many instances plotted in such small areas that it was difficult identifying areas of high instances. Therefore a heatmap was used instead which had a colour scale of transparent to opaque red according to the number of instances occuring in particular areas. This map clearly shows there are multiple patches of salford which contain particularly high instances of anti-social behaviour.



6. Database security

Besides the methods of security which have been applied in the database already, there are other recommended steps which can be taken to improve the security of the database. It is possible to encrypt data to protect it from any threats or attempts to access, particularly if this data is deemed sensitive (Connolly and Begg, 2015). SQL Server offers multiple functions which can be used to encrypt data, such as ‘ENCRYPTBYKEY’ and ‘ENCRYPTBYPASSPHRASE’ (Mukherjee, 2019). Besides tools within the database, it is also recommended that external security measures are also taken (Mukherjee, 2019, Elmasri and Navathe, 2017). For example, the hardware used to run SQL server must be fault tolerant. Should one of the components fail the server can continue to run without failure. A potential option is using Redundant Array of Independent Disks (RAID) which allows data to be segmented and distributed across multiple disks (Connolly and Begg, 2015). Additional external security can involve ensuring the operating system used is up-to-date with the latest upgrades to security and that there is an operational firewall in place (Connolly and Begg, 2015). Finally, it is recommended that the hardware has physical security, such as locking the server hardware away and restricting access (Connolly and Begg, 2015).

7. Database backup

Backing up and recovering a database is another form of security. Should a failure occur where the database can no longer be used, a recent back up version can be restored (Connolly and Begg, 2015).

8. Data Science/Business Intelligence Techniques

Due to the amount of historical crime data, number of different types of reports created and the amount of areas data is gathered for across the UK, the client may benefit from using data warehousing. Data warehouses are defined as “a subject-orientated, integrated, time-variant, and non-volatile collection of data in support of management’s decision making process” (Connolly and Begg, 2015). This means that data can be stored according to subjects (e.g. the type of reported data, the overall area, etc.), all data is stored in a consistent format, historical data to present day data can be stored with time references, and data is updated regularly allowing new data to be integrated with old data, creating a fuller picture. This would allow users the flexibility and ease of accessing data from different time periods, to different areas, to different crime report types from one central server (Connolly and Begg, 2015). However, creating a data warehouse can have some drawbacks, with a large number of resources required to transform and load the data into the server, to homogenise the data into the same format if historical data collection is not consistent, larger disk space often needed to hold the large amounts of information which continues to increase and the high maintenance level of retaining consistency in the data (Connolly and Begg, 2015). It may be worth the investment, however, when considering the opportunities having access to large amounts of data can bring. Having large amounts of historical data stored can allow for data mining techniques to be applied (Connolly and Begg, 2015). For example, classification techniques can be used to identify patterns in criminal activity, such as where crime types are likely to occur, which could help investigators to understand what drives high levels of particular crime types and therefore lead to better interventions to be put in place.

9. Conclusion

Using a database to store crime report data can allow maintenance of data integrity, accuracy, and concurrency. It can also allow multiple users to use the data for future research through allowing users to create their own required combinations of attributes and calculated fields, without impacting the stored data. In the future, the organisation could store historical data and future data, creating a data warehouse and allowing data mining techniques to be applied.

References

BRIONES, K. 2018. A Guide to Young Lives Rounds 1 to 5 constructed files. *Young Lives Technical Note,* 48**,** 1-31.

CODD, E. F. 1972. Further normalization of the data base relational model. *Data base systems,* 6**,** 33-64.

CONNOLLY, T. & BEGG, C. 2015. Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.

CRIVELLO, G., MORROW, V. & WILSON, E. 2013. Young Lives longitudinal qualitative research: A guide for researchers.

CROOKSTON, B. T., FORSTE, R., MCCLELLAN, C., GEORGIADIS, A. & HEATON, T. B. 2014. Factors associated with cognitive achievement in late childhood and adolescence: the Young Lives cohort study of children in Ethiopia, India, Peru, and Vietnam. *BMC pediatrics,* 14**,** 1-9.

CUETO, S., FELIPE, C. & LEÓN, J. 2020. Predictors of school dropout across Ethiopia, India, Peru and Vietnam.

DERCON, S. & KRISHNAN, P. 2009. Poverty and the psychosocial competencies of children: evidence from the young lives sample in four developing countries. *Children Youth and Environments,* 19**,** 138-163.

ELMASRI, R. & NAVATHE, S. B. 2017. Fundamentals of Database Systems 7 th Edition. Pearson Harlow.

FRIEDMAN, J. & ROSENBAUM, D. P. 1988. Social control theory: The salience of components by age, gender, and type of crime. *Journal of Quantitative Criminology,* 4**,** 363-381.

IYER, P., AZUBUIKE, O. B. & ROLLESTON, C. 2017. Young Lives School Survey, 2016-17: Evidence from Vietnam. *Country Report, Oxford: Young Lives*.

MORROW, V. & PELLS, K. 2012. Integrating children's human rights and child poverty debates: Examples from young lives in Ethiopia and India. *Sociology,* 46**,** 906-920.

MUKHERJEE, S. 2019. SQL Server Development Best Practices. *International Journal of Innovative Research in Computer and Communication Engineering*, 10.

NGUYEN, H. T. 2019. Ethnic gaps in child education outcomes in Vietnam: an investigation using Young Lives data. *Education Economics,* 27**,** 93-111.

PELLS, K. 2011a. Poverty and gender inequalities: evidence from Young Lives.

PELLS, K. 2011b. Poverty, risk and families' responses: Evidence from Young Lives.

ROLLESTON, C. & IYER, P. 2019. Beyond the basics: Access and equity in the expansion of post-compulsory schooling in Vietnam. *International Journal of Educational Development,* 66**,** 223-233.

ROLLESTON, C. & KRUTIKOVA, S. 2014. Equalising opportunity? School quality and home disadvantage in Vietnam. *Oxford review of education,* 40**,** 112-131.

SHNEIDERMAN, B., PLAISANT, C., COHEN, M. S., JACOBS, S., ELMQVIST, N. & DIAKOPOULOS, N. 2016. *Designing the user interface: strategies for effective human-computer interaction*, Pearson.