2020

Data Science Foundation

Module number: DALT7002

Student Number: 19132761

MSc Course: MSc in Data Analytics

Word count: 1567

Content

Recording link

https://drive.google.com/file/d/15v1Wpm_2fhMSwgRlEjhRWPUVx7d5k3jW/view?usp=sharing

1. Data selection and cleansing

The steps in the data science process are followed as follows

- Obtaining the data
- Scrubbing the data
- Exploring the data
- Modelling the data
- Interpreting the data

For obtaining the data, the skills required are the database management skills when the dataset obtained is the unstructured data. The database management systems such as MySQL, SQLite and MongoDB can be used for unstructured data.

For scrubbing the data, R programming can be used on the raw dataset.

For the given problem, SQLite is used for obtaining the data and R programming is used for scrubbing the data.

The first step in data science is collecting the data for the given problem. All source dataset in this report have been collected from The United Kingdom Parliament and The Office for National Statistics (ONS). For example, the broadband speed dataset originated from the House of Commons Library of the United Kingdom Parliament which publishes many impartial analysis, statistical research to support staff legislation, develop policy. Meanwhile, the house prices datasets have been downloaded from ONS who the United Kingdom's largest producer of official statistics and its recognized national statistical institute are. It takes the responsibility to promote and protect the publication of official statistics for public good purposes.

- The dataset of house prices will be downloaded from 2 sources:
    a. **House Median**: The data from Office of National Statistics (ONS): Median price paid by ward, England and Wales, year ending Dec 1995 to year ending Dec 2019". https://www.ons.gov.uk/peoplepopulationandcommunity/housing/datasets/medianpricepaidbywardhpssadataset37
    b. **House Paid**: HM Land Registry n.d, HM Land Registry Open Data: Price Paid Data (the criteria search only for Town/City: Oxford) http://landregistry.data.gov.uk/app/ppd
- **Broadband**: The dataset of broadband coverage and speeds (year of 2017) will be downloaded from House of Commons Library (United Kingdom Parliament). https://commonslibrary.parliament.uk/research-briefings/cbp-8200/

- **Postcode:** The dataset of postcode will be downloaded from Office of National Statistics (ONS): Postcode to Parish to Ward to Local Authority District Lookup in England and Wales.
  https://geoportal.statistics.gov.uk/datasets/c4aeb11ff5b045018b7340e807d645cb

The next step is to cleanse the data collected. The data cleansing process involves checking the missing values, invalid entries in the collected data and data range problems. If any row or column does not seem to be valid data, then the invalid entries need to be modified with the default entries to make it the valid entry. Merging and splitting of columns can also be done to make the dataset valid (KDNuggets, 2016). Moreover, there is some redundancy information on each file that is required to remove before importing to the database. Hence, the process of data cleansing will be divided into 2 stages.

1.1. Excel

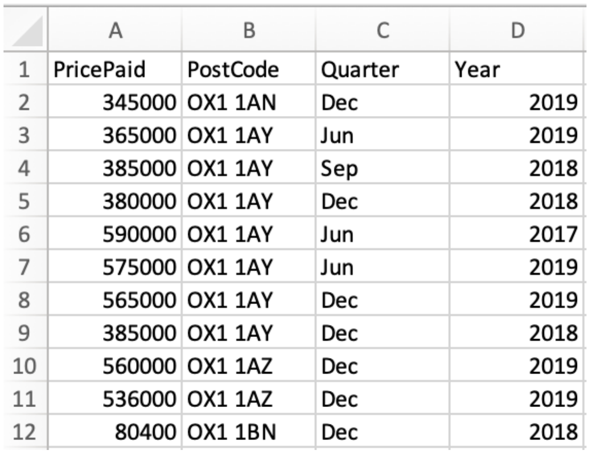
The first stage is using excel and its function to select the sheets/data which are needed.

House Median

In House Median file, use the 1a sheet and delete the first 5 rows unnecessary.

House Paid

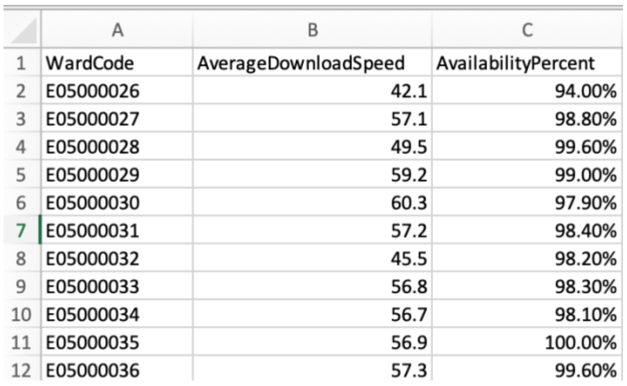
- Keep column *deed_date, postcode, price_paid*
- Rename column name: Date, PostCode, PricePaid.
- Extract Quarter and Year from Date by formulas:
  Quarter: = INDEX({"Mar","Jun","Sep","Dec"},ROUNDUP(MONTH(A2)/3,0))
  Year: = TEXT(A2,"YYYY")
- Create **housepaid** which contains: *PostCode*, *Quarter*, *Year*, *PricePaid*.



	A	B	C	D
1	PricePaid	PostCode	Quarter	Year
2	345000	OX1 1AN	Dec	2019
3	365000	OX1 1AY	Jun	2019
4	385000	OX1 1AY	Sep	2018
5	380000	OX1 1AY	Dec	2018
6	590000	OX1 1AY	Jun	2017
7	575000	OX1 1AY	Jun	2019
8	565000	OX1 1AY	Dec	2019
9	385000	OX1 1AY	Dec	2018
10	560000	OX1 1AZ	Dec	2019
11	536000	OX1 1AZ	Dec	2019
12	80400	OX1 1BN	Dec	2018

BroadBand

- Use the Ward Data sheet and delete all the information rows which are not needed.
- Keep and Rename column name: WardCode, AverageDownloadSpeed, AvailabilityPercent.



	A	B	C
1	WardCode	AverageDownloadSpeed	AvailabilityPercent
2	E05000026	42.1	94.00%
3	E05000027	57.1	98.80%
4	E05000028	49.5	99.60%
5	E05000029	59.2	99.00%
6	E05000030	60.3	97.90%
7	E05000031	57.2	98.40%
8	E05000032	45.5	98.20%
9	E05000033	56.8	98.30%
10	E05000034	56.7	98.10%
11	E05000035	56.9	100.00%
12	E05000036	57.3	99.60%

1.2. R programming

The next stage is using R programming language to clean, reformat the data and check the validity of data such as null value and restructure the format of data.

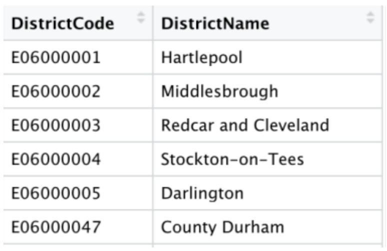
House Median

From HouseMedian dataset, extract 3 tables/dataframes: **district, ward** and **housemedian**.

- Import HouseMedian dataset to R
- Rename 4 column names which are used: *WardCode, WardName, DistrictCode, DistrictName*
- Extract **district** (include *DistrictCode, DistrictName*), **ward** (include *DistrictCode, WardCode, WardName*) and remove duplicate.
- Format *Year ending (Quarter) (Year)* column name to *QuarterYear*. Then create Quarter and Year vector replicate n times corresponding to n row dataset (exclude first 4 columns that do not contain data)
- Create a vector that replicate n times corresponding to n row of dataset (exclude first 4 columns that do not contain data)
- Create a vector that contains all of house median value
- Transpose HouseMedian
- Create WardCode vector is repeated (QuarterYear) times and Median contains all of house median value.

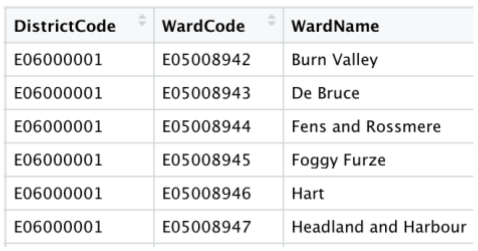
- If median value is not a number, then convert to 0.
- Create **housemedian** contains *WardCode, Quarter, Year, Median*.

district



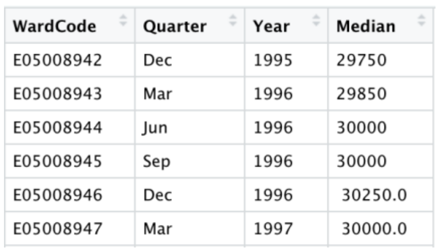
DistrictCode	DistrictName
E06000001	Hartlepool
E06000002	Middlesbrough
E06000003	Redcar and Cleveland
E06000004	Stockton-on-Tees
E06000005	Darlington
E06000047	County Durham

ward



DistrictCode	WardCode	WardName
E06000001	E05008942	Burn Valley
E06000001	E05008943	De Bruce
E06000001	E05008944	Fens and Rossmere
E06000001	E05008945	Foggy Furze
E06000001	E05008946	Hart
E06000001	E05008947	Headland and Harbour

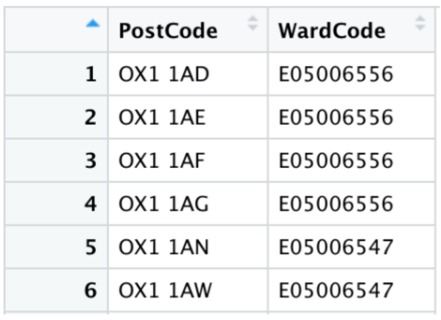
housemedian



WardCode	Quarter	Year	Median
E05008942	Dec	1995	29750
E05008943	Mar	1996	29850
E05008944	Jun	1996	30000
E05008945	Sep	1996	30000
E05008946	Dec	1996	30250.0
E05008947	Mar	1997	30000.0

Post Code

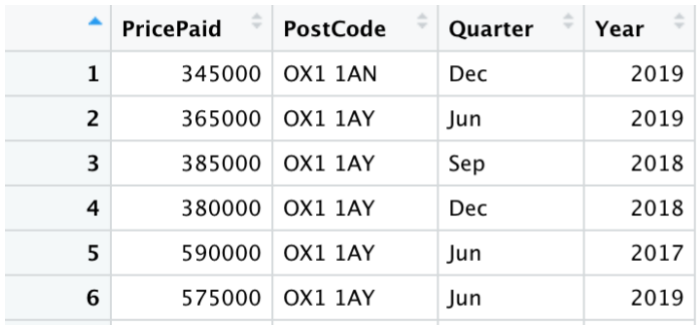
- Import PostCode dataset in R
- Select only Oxfordshire
- Rename column name: PostCode, WardCode
- Extract **postcode** which contains: *PostCode, WardCode*



	PostCode	WardCode
1	OX1 1AD	E05006556
2	OX1 1AE	E05006556
3	OX1 1AF	E05006556
4	OX1 1AG	E05006556
5	OX1 1AN	E05006547
6	OX1 1AW	E05006547

House Paid

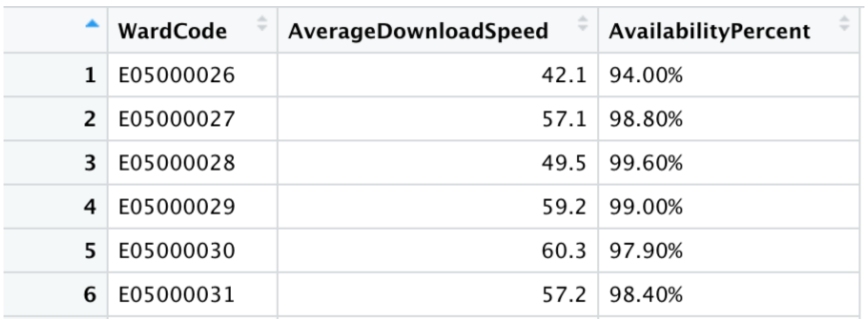
- Import HousePaid dataset in R



	PricePaid	PostCode	Quarter	Year
1	345000	OX1 1AN	Dec	2019
2	365000	OX1 1AY	Jun	2019
3	385000	OX1 1AY	Sep	2018
4	380000	OX1 1AY	Dec	2018
5	590000	OX1 1AY	Jun	2017
6	575000	OX1 1AY	Jun	2019

Broadband

- Import Broadband dataset in R



	WardCode	AverageDownloadSpeed	AvailabilityPercent
1	E05000026	42.1	94.00%
2	E05000027	57.1	98.80%
3	E05000028	49.5	99.60%
4	E05000029	59.2	99.00%
5	E05000030	60.3	97.90%
6	E05000031	57.2	98.40%

2. Legal and/or ethical issues

The necessary legal steps have to be followed while collecting and using the data for the data analysis process. The aim of various legal standards and acts is to protect the personal data. While collecting the data for research purpose, the researchers need to follow the following recommendations:

- The data which contains the personal information should not be collected if it is irrelevant for the research process.
- The researcher should avoid the personally identifiable information in the dataset if possible.
- The researcher should get the consent of the owner of the data before using the data for research purposes.
- If the research dataset requires the personally identifiable information, perform the de identification process as soon as collected.
- The dataset should not be transmitted without encrypting the data on the public internet.
- The device which contains the dataset need to be secured (Ulikool, 2020).

Following are the ethical issues involved in collecting and using the data for research process.

- Informed Consent is getting the approval of the person whose data is used for the research purpose.
- Beneficence is the responsibility of the researcher to benefit the society by doing the research with the gathered data and it should not create harm to anyone.
- The researcher has to maintain the anonymity of the persons involved with the data if possible. If anonymity is not possible, the researcher should ensure the confidentiality of the data when the persons involved.
- The privacy of the individual also needs to be respected when performing the research on the dataset related to an individual (Fouka, 2019)

3. Structured and semi-structured data

Structured data is a type of data where the records are identifiable for efficient analysis. It has been combined into a formatted storage that is commonly a database. It comprises of all data which can be saved in database SQL as a table with records and attributes. They have identification keys and can easily be mapped into assigned fields. This is the common and simple way for managing the data. The common example for structured data is SQL.

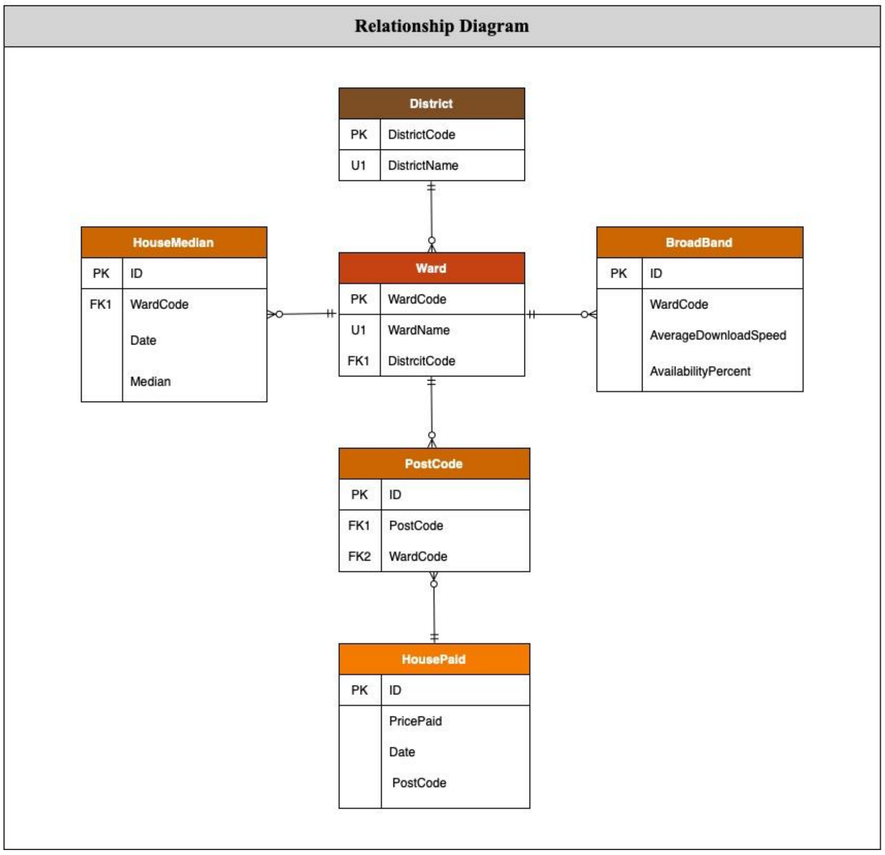
Semi-structured data is a type of data that does not store in a relational database but that contains some structure properties that make it easy for analysing the data. By using some standard process, this data can be stored in the relation database, but it follows the Semi structured form to save the storage space. The common example for semi structured data is the XML data.

The SQL data which is structured can be easily organized and it follows a structured format. This is used for easy manipulation and extraction of data. The structured data can be easily used by the machine learning algorithms.

4. Data model and implementation

4.1.Normalisation of data to 3NF

The table is said to be in third normal form when there is no partial dependency and transitive dependency among the columns of the table. The partial dependency occurs when a composite attribute is used to uniquely identify the non-key attributes. The transitive dependency occurs when there is a chain of dependency between the attributes of the table. In the given dataset, the dependency can be removed and converted into third normal form by splitting the dataset into five tables namely District, Ward, HouseMedian, HousePrice and Broadband.



4.2. Appropriate design of SQL database tables

- Table **District**: Primary is DistrictCode which is used for uniquely identifying this row in the table.

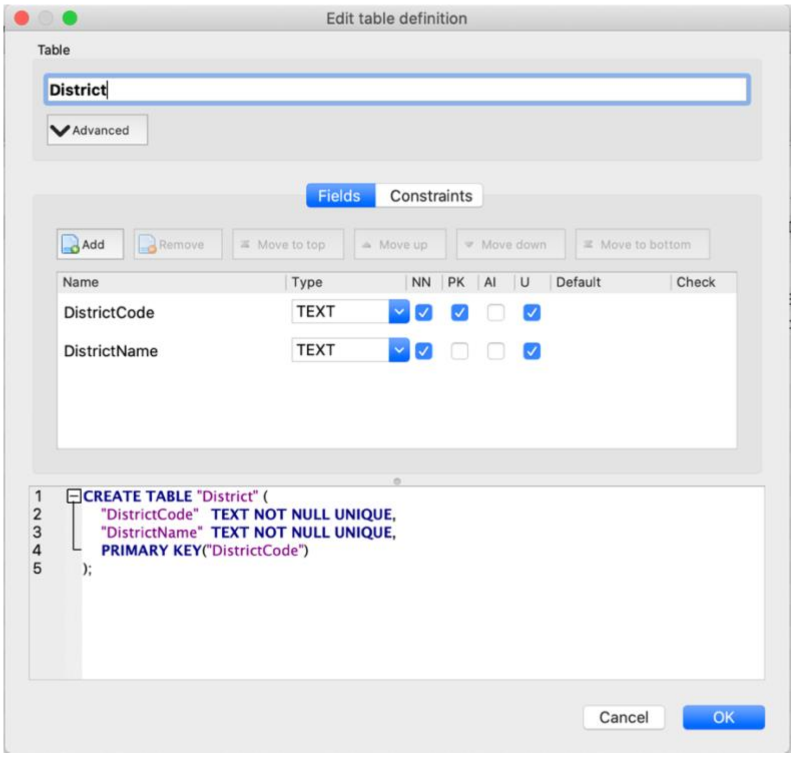


Table: District

Name	Type	NN	PK	AI	U	Default	Check
DistrictCode	TEXT	✓	✓		✓		
DistrictName	TEXT	✓			✓		

```
CREATE TABLE "District" (
    "DistrictCode"   TEXT NOT NULL UNIQUE,
    "DistrictName"   TEXT NOT NULL UNIQUE,
    PRIMARY KEY("DistrictCode")
);
```

- Table **PostCode**: Foreign keys are PostCode and WardCode. Primary key is ID (auto increment). Here, DistrictCode is the foreign key since it references the District table. The WardCode is used to uniquely identify the row in this table.

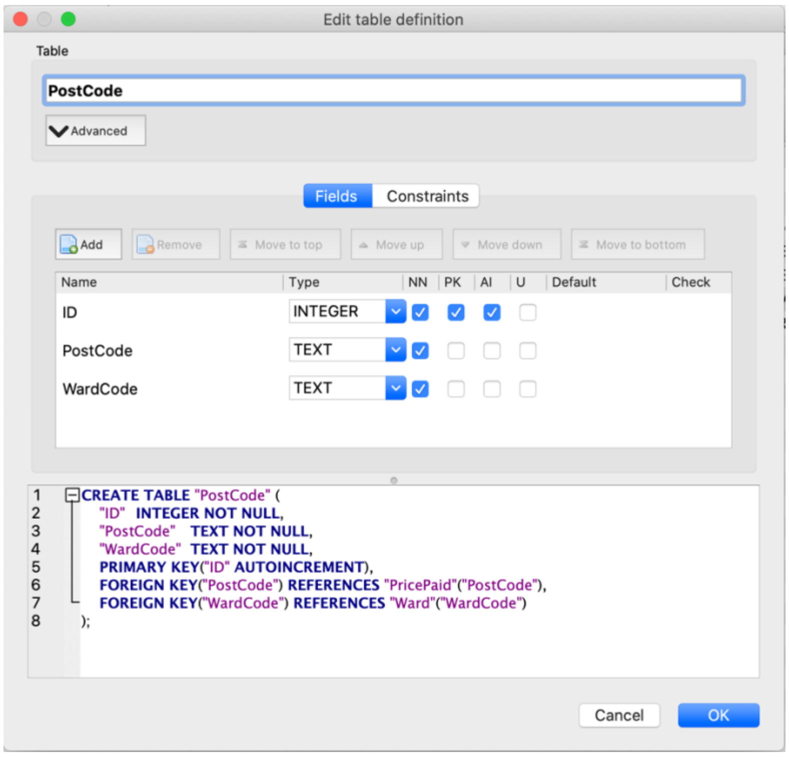


Table: PostCode

Name	Type	NN	PK	AI	U	Default	Check
ID	INTEGER	✓	✓	✓	☐		
PostCode	TEXT	✓	☐	☐	☐		
WardCode	TEXT	✓	☐	☐	☐		

```
CREATE TABLE "PostCode" (
    "ID"  INTEGER NOT NULL,
    "PostCode"   TEXT NOT NULL,
    "WardCode"  TEXT NOT NULL,
    PRIMARY KEY("ID" AUTOINCREMENT),
    FOREIGN KEY("PostCode") REFERENCES "PricePaid"("PostCode"),
    FOREIGN KEY("WardCode") REFERENCES "Ward"("WardCode")
);
```

- Table **Ward**: Foreign Key is DistrictCode and Primary Key is WardCode. Here, DistrictCode is the foreign key since it references the District table. The WardCode is used to uniquely identify the row in this table.

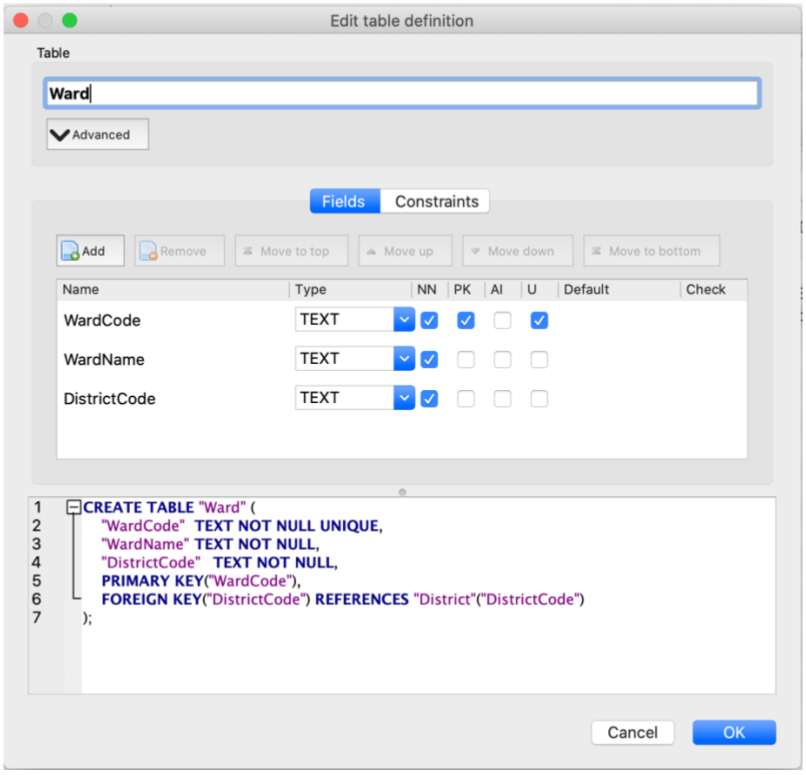


Table: Ward

Name	Type	NN	PK	AI	U	Default	Check
WardCode	TEXT	✓	✓	☐	✓		
WardName	TEXT	✓	☐	☐	☐		
DistrictCode	TEXT	✓	☐	☐	☐		

```
CREATE TABLE "Ward" (
    "WardCode"  TEXT NOT NULL UNIQUE,
    "WardName" TEXT NOT NULL,
    "DistrictCode"  TEXT NOT NULL,
    PRIMARY KEY("WardCode"),
    FOREIGN KEY("DistrictCode") REFERENCES "District"("DistrictCode")
);
```

- Table **HouseMedian**: Foreign Key is Wardcode. Primary Key is ID (auto increment). In this table, WardCode is foreign key since it references WardCode in the Ward table.

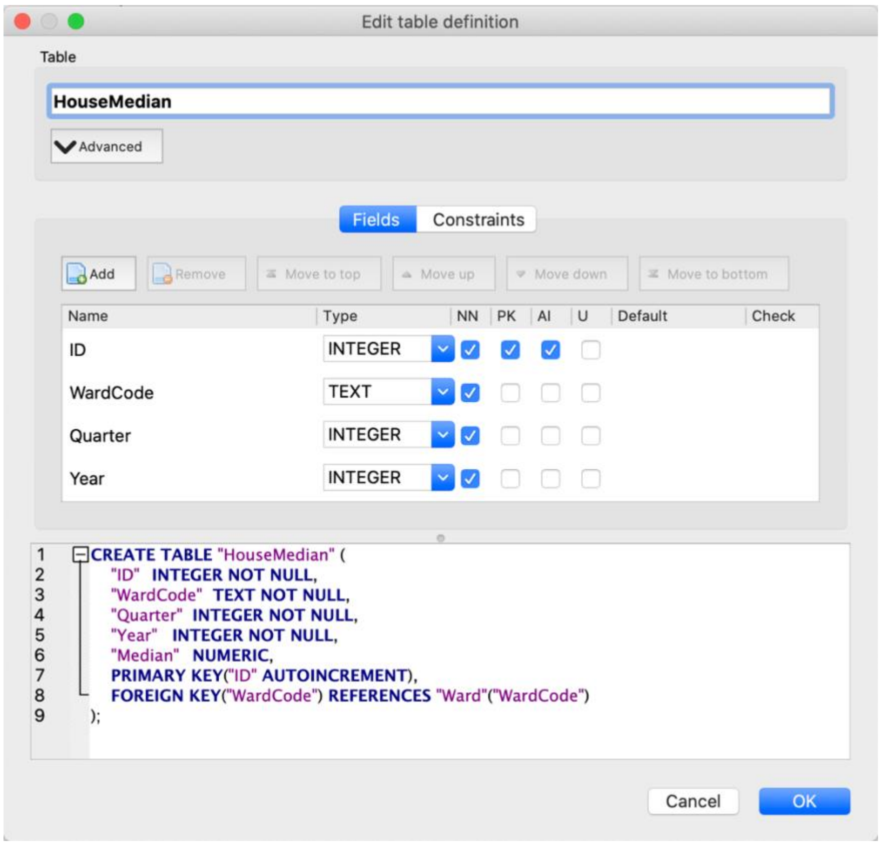


Table: HouseMedian

Name	Type	NN	PK	AI	U	Default	Check
ID	INTEGER	✓	✓	✓			
WardCode	TEXT	✓					
Quarter	INTEGER	✓					
Year	INTEGER	✓					

```
CREATE TABLE "HouseMedian" (
    "ID"  INTEGER NOT NULL,
    "WardCode"  TEXT NOT NULL,
    "Quarter"  INTEGER NOT NULL,
    "Year"   INTEGER NOT NULL,
    "Median"  NUMERIC,
    PRIMARY KEY("ID" AUTOINCREMENT),
    FOREIGN KEY("WardCode") REFERENCES "Ward"("WardCode")
);
```

- Table **HousePaid**: Foreign Key is PostCode. Primary Key is ID (auto increment).

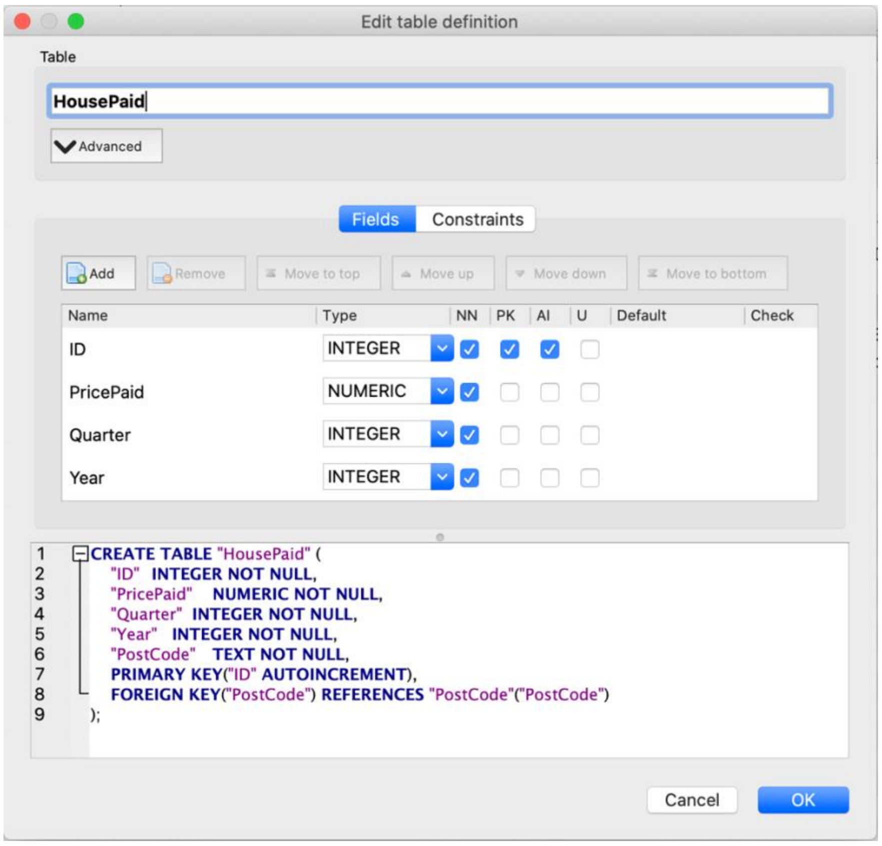
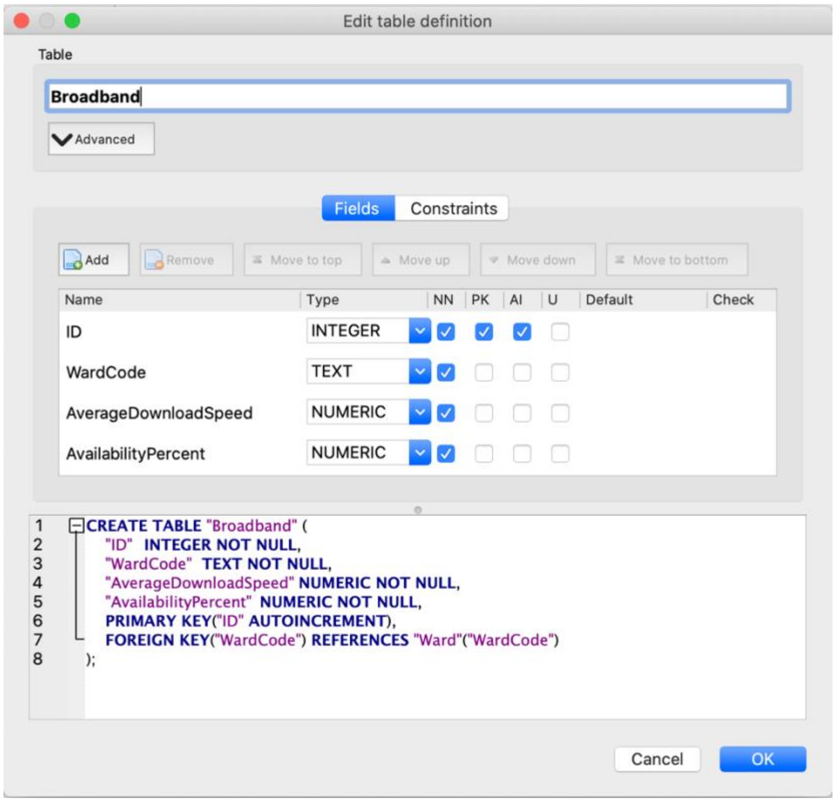


Table: HousePaid

Name	Type	NN	PK	AI	U	Default	Check
ID	INTEGER	✓	✓	✓			
PricePaid	NUMERIC	✓					
Quarter	INTEGER	✓					
Year	INTEGER	✓					

```
CREATE TABLE "HousePaid" (
    "ID"  INTEGER NOT NULL,
    "PricePaid"   NUMERIC NOT NULL,
    "Quarter"  INTEGER NOT NULL,
    "Year"   INTEGER NOT NULL,
    "PostCode"   TEXT NOT NULL,
    PRIMARY KEY("ID" AUTOINCREMENT),
    FOREIGN KEY("PostCode") REFERENCES "PostCode"("PostCode")
);
```

- Table **Broadband**: Foreign Key is WardCode, Primary Key is ID (auto increment).



Name	Type	NN	PK	AI	U	Default	Check
ID	INTEGER	✓	✓	✓			
WardCode	TEXT	✓					
AverageDownloadSpeed	NUMERIC	✓					
AvailabilityPercent	NUMERIC	✓					

```
CREATE TABLE "Broadband" (
    "ID"   INTEGER NOT NULL,
    "WardCode"   TEXT NOT NULL,
    "AverageDownloadSpeed" NUMERIC NOT NULL,
    "AvailabilityPercent"  NUMERIC NOT NULL,
    PRIMARY KEY("ID" AUTOINCREMENT),
    FOREIGN KEY("WardCode") REFERENCES "Ward"("WardCode")
);
```

5. Connect and insert data to Database

```
# Connect to DB
conn <- dbConnect(RSQLite::SQLite(), "Database.db")


# Insert Table
dbGetQuery(conn, "SELECT count(*) FROM (table)")
dbWriteTable(conn, name = "(table)", value = (dataframe), append = TRUE)


# Delete values
dbGetQuery(conn, "delete from (table)")


# Disconnect
dbDisconnect(conn)
```

6. Data model and implementation

Question 3: The average prices of houses in 2018 in Cherwell

```
# Question 3
dbGetQuery(conn, "select avg(Median) from HouseMedian where year = '2018' and
WardCode in (select WardCode from Ward where DistrictCode = (select
           DistrictCode from District where DistrictName = 'Cherwell'))")
```

```
##   avg(Median)
## 1      280790
```

Question 4: The average increase in prices (in percent) between 2017 and 2018 in Headington Ward

```
# Question 4
dbGetQuery(conn,"select     w.WardName, hm2017.Year as Year2017,
(sum(hm2017.Median)/4) as AvgPrice2017, hm2018.Year as Year2018,
(sum(hm2018.Median)/4) as AvgPrice2018, (sum(hm2018.Median)/4) -
(sum(hm2017.Median)/4) as ChangeInPrice,((sum(hm2018.Median)/4) -
(sum(hm2017.Median)/4))*100/(sum(hm2017.Median)/4)
as 'ChangeInPercent(%)'
    from HouseMedian hm2017
    inner join HouseMedian hm2018 on hm2017.WardCode = hm2018.WardCode
    inner join Ward w on hm2017.WardCode = w.WardCode
    where w.WardName = 'Headington' and hm2017.Year = '2017' and hm2018.Year = '2018'
    group by hm2017.Year, hm2018.Year,w.WardName")
```

```
##     WardName Year2017 AvgPrice2017 Year2018 AvgPrice2018 ChangeInPrice
## 1 Headington     2017      1091500     2018      1347125        255625
##   ChangeInPercent(%)
## 1                 23
```

Question 5: A ward in Oxford which has the highest house price in Mar 2018

```
# Question 5
dbGetQuery(conn,"select  la.DistrictName, w.WardName,hp.Year, hp.Quarter,
max(hp.PricePaid) from HousePaid hp
inner join PostCode pc on pc.PostCode = hp.PostCode
inner join Ward w on w.WardCode = pc.WardCode
inner join District la on la.DistrictCode = w.DistrictCode
where la.DistrictName in ('Oxford','Cherwell','South Oxfordshire',
'Vale of White Horse','West Oxfordshire')
and hp.Quarter = 'Mar' and hp.Year = '2018'
group by hp.Quarter, hp.Year, la.DistrictName, w.WardName
order by max(hp.PricePaid) desc
limit 1")
```

```
##   DistrictName WardName Year Quarter max(hp.PricePaid)
## 1       Oxford    North 2018     Mar           6000000
```

Question 6: A ward in Oxford which has the lowest house price in Dec 2019.

```
# Question 6
dbGetQuery(conn,"select  la.DistrictName, w.WardName,hp.Year, hp.Quarter,
max(hp.PricePaid) from HousePaid hp
inner join PostCode pc on pc.PostCode = hp.PostCode
inner join Ward w on w.WardCode = pc.WardCode
inner join District la on la.DistrictCode = w.DistrictCode
where la.DistrictName in ('Oxford','Cherwell','South Oxfordshire',
'Vale of White Horse','West Oxfordshire')
and hp.Quarter = 'Dec' and hp.Year = '2019'
group by hp.Quarter, hp.Year, la.DistrictName, w.WardName
order by max(hp.PricePaid)
limit 1")
```

```
##   DistrictName       WardName Year Quarter max(hp.PricePaid)
## 1       Oxford Blackbird Leys 2019     Dec            305000
```

Question 7: Broadband speed (average downland) and (superfast) broadband availability (%) in Headington Ward.

```
# Question 7
dbGetQuery(conn,"select w.WardName, bs.* from Broadband bs
inner join Ward w on bs.WardCode = w.WardCode
where w.WardName = 'Headington'")
```

```
##     WardName     ID  WardCode AverageDownloadSpeed AvailabilityPercent
## 1 Headington 122656 E05006551                 56.6              99.70%
```

Question 8: A ward which has max average download speed and the ward which has max broadband availability percent

```
# Question 8
dbGetQuery(conn,"select  max(bs.AverageDownloadSpeed ), w.WardName
from Broadband bs inner join Ward w on bs.WardCode = w.WardCode")
```

```
##   max(bs.AverageDownloadSpeed )          WardName
## 1                         472.8 Upper Lune Valley
```

```
dbGetQuery(conn,"select  max(bs.AvailabilityPercent ), w.WardName
from Broadband bs inner join Ward w on bs.WardCode = w.WardCode")
```

```
##   max(bs.AvailabilityPercent ) WardName
## 1                       99.90% Parsloes
```

References

Fouka, G. &. (2019). What are the major ethical issues in conducting research? *Health Science Journal*.

KDNuggets. (2016). *Data Science Process*. Retrieved from https://www.kdnuggets.com/2016/03/data-science-proces (Fouka, n.d.)s.html, pp. 1-1.

Ulikool, T. (2020). Retrieved from Legal and Ethical Issues: https://sisu.ut.ee/rdm_course1/acts-law-and-research-ethics

Appendix

```
#install.packages("DBI")
#install.packages("odbc")
library(haven)
library(dplyr)

library(odbc)
library(readxl)



##### House Paid #####
housepaid <- read.csv("/Users/maido/Desktop/Dataset/Data Science/HousePaid.csv")

##### BroadBand #####
broadband <- read.csv("/Users/maido/Desktop/Dataset/Data Science/Broadband.csv")




##### House Median #####
HM <- read_excel("/Users/maido/Desktop/Dataset/Data Science/HouseMedian.xlsx")

# Rename the column name
names(HM)[names(HM) == "Ward name"] <- "WardName"
names(HM)[names(HM) == "Ward code"] <- "WardCode"
names(HM)[names(HM) == "Local authority code"] <- "DistrictCode"
names(HM)[names(HM) == "Local authority name"] <- "DistrictName"


# Create District table
district <- subset(HM, select=c(DistrictCode, DistrictName))
# Remove duplicate
district <- district %>% distinct(DistrictCode, .keep_all = TRUE)
```

```
# Create Ward table
ward <- subset(HM, select=c(DistrictCode, WardCode, WardName))
# Remove duplicate
ward <- ward %>% distinct(WardCode, .keep_all = TRUE)


# Remove "Year ending " column name
for ( col in 1:ncol(HM)){
  colnames(HM)[col] <-  sub("Year ending ", "",
                                colnames(HM)[col])}
# Remove " "
names(HM) <- gsub("\\ ", "", names(HM))

# Extract Quarter and Year
Quarter <- substring(names(HM[,5:101]),1,3)
Year <- substring(names(HM[,5:101]),4,7)

# Create vector Quarter and Year
Quarter <- rep(Quarter, times = nrow(HM))
Year <- rep(Year, times = nrow(HM))

# Transpose HM
HMtran <- t(HM)

# Create a vector that replicate n times corresponding to n row of HMtran (exclude first 4 columns th
at not contain data)
WardCode <- rep(c(HMtran[3,1:ncol(HMtran)]),times=nrow(HMtran)-4)

# Create a vector that contains all of house median value
Median <- as.vector(HMtran[5:nrow(HMtran),])

# Create housemedian dataframe
housemedian <- data.frame(WardCode,Quarter,Year,Median)




##### Postcode #####
postcode <- read.csv("/Users/maido/Desktop/Dataset/Data Science/Postcode.csv", header = TRUE)
#head(postcode)

# Select only Oxfordshire
postcode <- postcode %>% filter(lad11nm == "Oxford")

# Rename the column name
names(postcode)[1] <- "PostCode"
names(postcode)[names(postcode) == "wd11cd"] <- "WardCode"

# Keep PostCode and WardCode
postcode <- subset(postcode, select = c(PostCode, WardCode))
```

```
##### Connect with Database ######

# Connect to DB
conn <- dbConnect(RSQLite::SQLite(), "Database.db")

# Insert Ward
dbGetQuery(conn, "SELECT count(*) FROM Ward")

##   count(*)
## 1        0

dbWriteTable(conn, name="Ward", value=ward, append=TRUE)

# Insert District
dbGetQuery(conn, "SELECT count(*) FROM District")

##   count(*)
## 1        0

dbWriteTable(conn, name="District", value=district, append=TRUE)

# Insert HouseMedian
dbGetQuery(conn, "SELECT count(*) FROM HouseMedian")

##   count(*)
## 1        0

dbWriteTable(conn, name="HouseMedian", value=housemedian, append=TRUE)

# Insert HousePaid
dbGetQuery(conn, "SELECT count(*) FROM HousePaid")

##   count(*)
## 1        0

dbWriteTable(conn, name="HousePaid", value=housepaid, append=TRUE)

# Insert PostCode
dbGetQuery(conn, "SELECT count(*) FROM PostCode")

##   count(*)
## 1        0

dbWriteTable(conn, name="PostCode", value=postcode, append=TRUE)

# Insert Broadband
dbGetQuery(conn, "SELECT count(*) FROM Broadband")

##   count(*)
## 1        0

dbWriteTable(conn, name="Broadband", value=broadband, append=TRUE)




##### Question 3-8 ######
```

```
# Question 3
dbGetQuery(conn, "select avg(Median) from HouseMedian where year = '2018' and
WardCode in (select WardCode from Ward where DistrictCode = (select
        DistrictCode from District where DistrictName = 'Cherwell'))")

##   avg(Median)
## 1      280790



# Question 4
dbGetQuery(conn,"select     w.WardName, hm2017.Year as Year2017,
(sum(hm2017.Median)/4) as AvgPrice2017, hm2018.Year as Year2018,
(sum(hm2018.Median)/4) as AvgPrice2018, (sum(hm2018.Median)/4) -
(sum(hm2017.Median)/4) as ChangeInPrice,((sum(hm2018.Median)/4) -
(sum(hm2017.Median)/4))*100/(sum(hm2017.Median)/4)
as 'ChangeInPercent(%)'
   from HouseMedian hm2017
   inner join HouseMedian hm2018 on hm2017.WardCode = hm2018.WardCode
   inner join Ward w on hm2017.WardCode = w.WardCode
   where w.WardName = 'Headington' and hm2017.Year = '2017' and hm2018.Year = '2018'
   group by hm2017.Year, hm2018.Year,w.WardName")

##     WardName Year2017 AvgPrice2017 Year2018 AvgPrice2018 ChangeInPrice
## 1 Headington     2017      1091500     2018      1347125        255625
##   ChangeInPercent(%)
## 1                 23



# Question 5
dbGetQuery(conn,"select  la.DistrictName, w.WardName,hp.Year, hp.Quarter,
max(hp.PricePaid) from HousePaid hp
inner join PostCode pc on pc.PostCode = hp.PostCode
inner join Ward w on w.WardCode = pc.WardCode
inner join District la on la.DistrictCode = w.DistrictCode
where la.DistrictName in ('Oxford','Cherwell','South Oxfordshire',
'Vale of White Horse','West Oxfordshire')
and hp.Quarter = 'Mar' and hp.Year = '2018'
group by hp.Quarter, hp.Year, la.DistrictName, w.WardName
order by max(hp.PricePaid) desc
limit 1")

##   DistrictName WardName Year Quarter max(hp.PricePaid)
## 1       Oxford    North 2018     Mar           6000000



# Question 6
dbGetQuery(conn,"select  la.DistrictName, w.WardName,hp.Year, hp.Quarter,
max(hp.PricePaid) from HousePaid hp
inner join PostCode pc on pc.PostCode = hp.PostCode
inner join Ward w on w.WardCode = pc.WardCode
inner join District la on la.DistrictCode = w.DistrictCode
where la.DistrictName in ('Oxford','Cherwell','South Oxfordshire',
'Vale of White Horse','West Oxfordshire')
```

```
and hp.Quarter = 'Dec' and hp.Year = '2019'
group by hp.Quarter, hp.Year, la.DistrictName, w.WardName
order by max(hp.PricePaid)
limit 1")

##   DistrictName       WardName Year Quarter max(hp.PricePaid)
## 1       Oxford Blackbird Leys 2019     Dec            305000



# Question 7
dbGetQuery(conn,"select w.WardName, bs.* from Broadband bs
inner join Ward w on bs.WardCode = w.WardCode
where w.WardName = 'Headington'")

##     WardName     ID  WardCode AverageDownloadSpeed AvailabilityPercent
## 1 Headington 168311 E05006551                 56.6              99.70%



# Question 8
dbGetQuery(conn," select  max(bs.AverageDownloadSpeed ), w.WardName from Broadband bs inn
er join Ward w on bs.WardCode = w.WardCode")

##     max(bs.AverageDownloadSpeed    WardName
## 1                            472.8 Upper     Lune Valley


dbGetQuery(conn," select  max(bs.AvailabilityPercent), w.WardName from BroadbandSpeed bs inn
er join Ward w on bs.WardCode = w.WardCode")

##     max(bs.AvailabilityPercent)            WardName
## 1                   99.90%                 Parsloes



##### Delete the values #####
dbGetQuery(conn, "delete from HouseMedian")

## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows

dbGetQuery(conn, "delete from Ward")

## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows

dbGetQuery(conn, "delete from HousePaid")

## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows

dbGetQuery(conn, "delete from Broadband")
```

```
## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows

dbGetQuery(conn, "delete from District")

## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows

dbGetQuery(conn, "delete from PostCode")

## Warning in result_fetch(res@ptr, n = n): SQL statements must be issued with
## dbExecute() or dbSendStatement() instead of dbGetQuery() or dbSendQuery().

## data frame with 0 columns and 0 rows



##### Disconnect ######
dbDisconnect(conn)
```