DA5020.Practicum 2

Chenyao Xiao, Yicheng Zhang and Isabella Motha

2022-11-13

Part 1 Customer Database Analysis

Here we need to create a new SQLite database.

```
mydb1 <- dbConnect(RSQLite::SQLite(), "my-db1.sqlite")
```

1

The commands for importing data to the database are listed in the sql file.

Here we found the headers are duplicated, thus we delete the first row to get the right Router_Info table.

```
dbListTables(mydb1)
```

```
## [1] "Router_Info" "customers"   "orders"
```

2

We can check the data by selecting first five rows from each table.

```
dbGetQuery(mydb1, 'SELECT * FROM customers LIMIT 5;')
```

```
##   CustomerID FirstName LastName              StreetAddress       City    State
## 1       1001    Wesley  Solomon 175 Green Clarendon Avenue     Toledo     Utah
## 2       1002      Dean  Bernard          956 Second Street Washington    Texas
## 3       1003   Valerie Schwartz           83 Milton Avenue      Miami Nebraska
## 4       1004   Loretta    Ewing        565 White First St.    Phoenix Delaware
## 5       1005      Noah     Boyd  35 South Green Oak Avenue    Detroit  Montana
##   ZipeCode    Telephone Purchases_Total
## 1    55933   9613675297               2
## 2    94778   6294571987               3
## 3    67569 679-609-0243               1
## 4    39399  239407-8990               2
## 5    59454  436931-2419               2
```

```
dbGetQuery(mydb1, 'SELECT * FROM orders LIMIT 5;')
```

```
##   OrderID CustomerID        SKU
## 1  201901       1001 BAS-08-1 C
## 2  201902       1002 BAS-24-1 C
## 3  201903       1003 BAS-48-1 C
## 4  201904       1004 ADV-24-10C
## 5  201905       1005 ADV-48-10F
##                                                    Description Cost
## 1                            Basic Switch  10/100/1000 BaseT 8 port  139
## 2                            Basic Switch 10/100/1000 BaseT 24 port   51
## 3                            Basic Switch 10/100/1000 BaseT 48 port  190
## 4                             Advanced Switch 10GigE Copper 24 port  247
## 5 Advanced Switch 10 GigE Copper/Fiber 44 port copper 4 port fiber  254
##   Year_Purchase
## 1          2019
## 2          2021
## 3          2021
## 4          2021
## 5          2019
```

```
dbGetQuery(mydb1, 'SELECT * FROM Router_Info LIMIT 5;')
```

```
##    RMAID OrderID                               Status     Step    Reason
## 1 707701  201901     SDJGDUVYIHOJZOOYMUNORAGYPLUN7UUSU Complete Incorrect
## 2 707702  201902                                    GA Complete Incorrect
## 3 707703  201903            FK2VI2NO4CDFK6M7BCZODTNICM Complete Incorrect
## 4 707704  201904             HQW3WX6CHP02E604FF76NF5LQ Complete Incorrect
## 5 707705  201905 J8ES48BYNXNICRDBW1E8L2V74LPI0UV7GOEIA Complete Incorrect
##   CustomerID
## 1       1001
## 2       1002
## 3       1003
## 4       1004
## 5       1005
```

By observing three tables, we can tell that they all have 499 observations and unique primary keys. Since they all have customerID ,OrderID and RMAID in its own table so each row of data should refer to a unique entity with its related in formations. We should be able to join all these three tables to find out the complete information about each observation. The primary key for customers table is customerID; for order table is OrderID; and for Router_info table is RMAID.The three tables contains different information are connected by customerID. The table orders and Router_info are connected by customerID and orderID. If we want to join them we could do like this:

```
# we join 3 tables like this:
sql<-"SELECT * FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID
JOIN Router_info ON Router_info.CustomerID = orders.CustomerID;"
joinresult<-dbGetQuery(mydb1,sql)
# check result:
str(joinresult)
```

```
## 'data.frame':    499 obs. of  21 variables:
##  $ OrderID        : int  201901 201902 201903 201904 201905 201906 201907 201908 201909 201910 ...
##  $ CustomerID     : chr  "1001" "1002" "1003" "1004" ...
##  $ SKU            : chr  "BAS-08-1 C" "BAS-24-1 C" "BAS-48-1 C" "ADV-24-10C" ...
```

```
##  $ Description    : chr  "Basic Switch  10/100/1000 BaseT 8 port" "Basic Switch 10/100/1000 BaseT 24
##  $ Cost           : int  139 51 190 247 254 74 87 129 167 159 ...
##  $ Year_Purchase  : int  2019 2021 2021 2021 2019 2020 2018 2019 2020 2021 ...
##  $ CustomerID     : int  1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...
##  $ FirstName      : chr  "Wesley" "Dean" "Valerie" "Loretta" ...
##  $ LastName       : chr  "Solomon" "Bernard" "Schwartz" "Ewing" ...
##  $ StreetAddress  : chr  "175 Green Clarendon Avenue" "956 Second Street" "83 Milton Avenue" "565 Wh
##  $ City           : chr  "Toledo" "Washington" "Miami" "Phoenix" ...
##  $ State          : chr  "Utah" "Texas" "Nebraska" "Delaware" ...
##  $ ZipeCode       : chr  "55933" "94778" "67569" "39399" ...
##  $ Telephone      : chr  "9613675297" "6294571987" "679-609-0243" "239407-8990" ...
##  $ Purchases_Total: int  2 3 1 2 2 2 2 7 2 6 ...
##  $ RMAID          : int  707701 707702 707703 707704 707705 707706 707707 707708 707709 707710 ...
##  $ OrderID        : chr  "201901" "201902" "201903" "201904" ...
##  $ Status         : chr  "SDJGDUVYIHOJZOOYMUNORAGYPLUN7UUSU" "GA" "FK2VI2NO4CDFK6M7BCZODTNICM" "HQW3
##  $ Step           : chr  "Complete" "Complete" "Complete" "Complete" ...
##  $ Reason         : chr  "Incorrect" "Incorrect" "Incorrect" "Incorrect" ...
##  $ CustomerID     : chr  "1001" "1002" "1003" "1004" ...
```

3

```
sql_cd1.3 ="SELECT COUNT(CustomerID) FROM orders WHERE COST > 200;"
dbGetQuery(mydb1, sql_cd1.3)
```

```
##   COUNT(CustomerID)
## 1               177
```

We create a query to get that 177 customers spent more than 200 dollars on the orders.

4

```
sql_cd1.4 ="SELECT *
FROM customers
JOIN orders ON orders.CustomerID = customers.CustomerID
JOIN Router_Info ON Router_Info.CustomerID = orders.CustomerID;"
df_join <-dbGetQuery(mydb1,sql_cd1.4)
head(df_join)
```

```
##   CustomerID FirstName LastName              StreetAddress       City
## 1       1001    Wesley  Solomon 175 Green Clarendon Avenue     Toledo
## 2       1002      Dean  Bernard          956 Second Street Washington
## 3       1003   Valerie Schwartz           83 Milton Avenue      Miami
## 4       1004   Loretta    Ewing        565 White First St.    Phoenix
## 5       1005      Noah     Boyd  35 South Green Oak Avenue    Detroit
## 6       1006     Trina     Wong      49 East Cowley Freeway    Yonkers
##          State ZipeCode    Telephone Purchases_Total OrderID CustomerID
## 1         Utah    55933   9613675297               2  201901       1001
## 2        Texas    94778   6294571987               3  201902       1002
## 3     Nebraska    67569 679-609-0243               1  201903       1003
```

```
## 4    Delaware    39399  239407-8990              2  201904       1004
## 5     Montana    59454  436931-2419              2  201905       1005
## 6 Connecticut    29552   6344811033              2  201906       1006
##          SKU                                                 Description
## 1 BAS-08-1 C                           Basic Switch  10/100/1000 BaseT 8 port
## 2 BAS-24-1 C                           Basic Switch 10/100/1000 BaseT 24 port
## 3 BAS-48-1 C                           Basic Switch 10/100/1000 BaseT 48 port
## 4 ADV-24-10C                            Advanced Switch 10GigE Copper 24 port
## 5 ADV-48-10F Advanced Switch 10 GigE Copper/Fiber 44 port copper 4 port fiber
## 6 ENT-24-10F                            Enterprise Switch 10GigE SFP+ 24 Port
##   Cost Year_Purchase  RMAID OrderID                               Status
## 1  139          2019 707701  201901     SDJGDUVYIHOJZOOYMUNORAGYPLUN7UUSU
## 2   51          2021 707702  201902                                    GA
## 3  190          2021 707703  201903           FK2VI2NO4CDFK6M7BCZODTNICM
## 4  247          2021 707704  201904            HQW3WX6CHP02E604FF76NF5LQ
## 5  254          2019 707705  201905 J8ES48BYNXNICRDBW1E8L2V74LPI0UV7GOEIA
## 6   74          2020 707706  201906                                  SO54
##       Step    Reason CustomerID
## 1 Complete Incorrect       1001
## 2 Complete Incorrect       1002
## 3 Complete Incorrect       1003
## 4 Complete Incorrect       1004
## 5 Complete Incorrect       1005
## 6 Complete Incorrect       1006
```

```
str(df_join)
```

```
## 'data.frame':    499 obs. of  21 variables:
##  $ CustomerID     : int  1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...
##  $ FirstName      : chr  "Wesley" "Dean" "Valerie" "Loretta" ...
##  $ LastName       : chr  "Solomon" "Bernard" "Schwartz" "Ewing" ...
##  $ StreetAddress  : chr  "175 Green Clarendon Avenue" "956 Second Street" "83 Milton Avenue" "565 Whi
##  $ City           : chr  "Toledo" "Washington" "Miami" "Phoenix" ...
##  $ State          : chr  "Utah" "Texas" "Nebraska" "Delaware" ...
##  $ ZipeCode       : chr  "55933" "94778" "67569" "39399" ...
##  $ Telephone      : chr  "9613675297" "6294571987" "679-609-0243" "239407-8990" ...
##  $ Purchases_Total: int  2 3 1 2 2 2 2 7 2 6 ...
##  $ OrderID        : int  201901 201902 201903 201904 201905 201906 201907 201908 201909 201910 ...
##  $ CustomerID     : chr  "1001" "1002" "1003" "1004" ...
##  $ SKU            : chr  "BAS-08-1 C" "BAS-24-1 C" "BAS-48-1 C" "ADV-24-10C" ...
##  $ Description    : chr  "Basic Switch  10/100/1000 BaseT 8 port" "Basic Switch 10/100/1000 BaseT 24
##  $ Cost           : int  139 51 190 247 254 74 87 129 167 159 ...
##  $ Year_Purchase  : int  2019 2021 2021 2021 2019 2020 2018 2019 2020 2021 ...
##  $ RMAID          : int  707701 707702 707703 707704 707705 707706 707707 707708 707709 707710 ...
##  $ OrderID        : chr  "201901" "201902" "201903" "201904" ...
##  $ Status         : chr  "SDJGDUVYIHOJZOOYMUNORAGYPLUN7UUSU" "GA" "FK2VI2NO4CDFK6M7BCZODTNICM" "HQW3W
##  $ Step           : chr  "Complete" "Complete" "Complete" "Complete" ...
##  $ Reason         : chr  "Incorrect" "Incorrect" "Incorrect" "Incorrect" ...
##  $ CustomerID     : chr  "1001" "1002" "1003" "1004" ...
```

We can first join two tables of customers and orders by CustomerID and then join Router_Info table as same. The data frame still contains 499 observations in total but 21 variables now, we use direct join here.

5

```
sql_cd1.5 ="SELECT orders.CustomerID, orders.OrderID, Description,
Purchases_Total, Year_Purchase
FROM customers
LEFT JOIN orders ON orders.CustomerID = customers.CustomerID
LEFT JOIN Router_Info ON Router_Info.CustomerID = orders.CustomerID;"
df_leftjoin <-dbGetQuery(mydb1,sql_cd1.5)
str(df_leftjoin)
```

```
## 'data.frame':    499 obs. of  5 variables:
##  $ CustomerID     : chr  "1001" "1002" "1003" "1004" ...
##  $ OrderID        : int  201901 201902 201903 201904 201905 201906 201907 201908 201909 201910 ...
##  $ Description    : chr  "Basic Switch  10/100/1000 BaseT 8 port" "Basic Switch 10/100/1000 BaseT 24
##  $ Purchases_Total: int  2 3 1 2 2 2 2 7 2 6 ...
##  $ Year_Purchase  : int  2019 2021 2021 2021 2019 2020 2018 2019 2020 2021 ...
```

In this query, we select some basic columns from the tables using left join.

6

The following SQL statement lists the number of customers in each city and their average cost total, sorted low to high (Only include cities with more than 8 customers):

```
sql_cd1.6 ="SELECT COUNT(Router_Info.CustomerID), City, AVG(Cost)
FROM customers
JOIN orders ON orders.CustomerID = customers.CustomerID
JOIN Router_Info ON Router_Info.CustomerID = orders.CustomerID
GROUP BY City
HAVING COUNT(Router_Info.CustomerID) > 8
ORDER BY COUNT(Router_Info.CustomerID);"
df_bycity <- dbGetQuery(mydb1,sql_cd1.6)
kable(df_bycity)
```

COUNT(Router_Info.CustomerID)	City	AVG(Cost)
9	Long Beach	138.5556
10	Detroit	169.9000
10	Kansas	180.7000
10	Santa Ana	158.7000
11	Washington	136.1818
15	Arlington	147.4667

We can see that there are 6 cities meeting our query standard. Arlington had most customers of 15, but Kansas had the highest avarage cost over 180 dollars.

7

```r
sql_cd1.7 ="SELECT Count (SKU) AS 'total_SKU',
COUNT ( DISTINCT Description ) AS 'total_description_types'
FROM orders
WHERE Cost>100;"
dbGetQuery(mydb1,sql_cd1.7)
```

```
##   total_SKU total_description_types
## 1       340                       8
```

We used the count and count distinctly to know that, among all of orders, those orders with cost more than 100, there are 8 distinct descriptions types and 340 total SKU exit in the order table.

8

```r
sql_cd1.8 = "SELECT Year_Purchase, MIN(Cost), MAX(Cost), SUM(Cost),
COUNT(OrderID) AS order_numbers
FROM orders
GROUP BY Year_Purchase
ORDER BY COUNT(OrderID) DESC;"
df_yearorders <- dbGetQuery(mydb1,sql_cd1.8)
kable(df_yearorders)
```

Year_Purchase	MIN(Cost)	MAX(Cost)	SUM(Cost)	order_numbers
2021	2	297	20472	133
2020	4	294	18758	130
2019	13	297	18649	118
2018	5	299	18607	118

The query gets a new table showing us order cost information each year, including minimum cost, maximum cost, sum and the order numbers. The table is display by descending order numbers. The year 2021 has the most orders.

9

```r
sql_cd1.9 = "SELECT FirstName, LastName,Status, Step,Reason, Purchases_Total
FROM customers
LEFT JOIN Router_info ON Router_info.CustomerID = customers.CustomerID
WHERE (customers.CustomerID BETWEEN 1001 AND 1021)
AND (Purchases_Total > 2)
AND (Purchases_Total < 10)
ORDER by Purchases_Total;"
df_purchase <- dbGetQuery(mydb1,sql_cd1.9)
dim(df_purchase)
```

```
## [1] 11  6
```

```
kable(head(df_purchase))
```

FirstName	LastName	Status	Step	Reason	Purchases_Total
Dean	Bernard	GA	Complete	Incorrect	3
Brandi	Rush	E8AGY2VAO23POO44POSVI5JZ1K	Complete	Defective	3
Marisa	Mc Cormick	C3	Complete	Incorrect	5
Vicki	Rowland	K5E2O4UHBIKBGKN	Complete	Incorrect	6
Sam	Austin	6CLQJ8CH6RZ1AO7V1	Complete	Incorrect	6
Glenn	Blackburn	SM91J6FEO7D2CCFAMJDER7I3EYVU7QHUEC2	Complete	Incorrect	7

For this query, we want to know the order status, order steps, reason and full name of customer who has a customerID between 1001 and 1021 and has a total purchase more than 2 but less than 10. It seems Dean Bernard and Brandi Rush have least purchases in these group of customers

10

```
sql_cd1.10 = "SELECT Year_Purchase, LastName,
COUNT(orders.OrderID) AS order_numbers, Description
FROM customers
LEFT JOIN orders ON orders.CustomerID = customers.CustomerID
WHERE Description LIKE '%Basic Switch%'
AND LastName LIKE 'B%'
GROUP BY Year_Purchase;"
kable(dbGetQuery(mydb1, sql_cd1.10))
```

Year_Purchase	LastName	order_numbers	Description
2018	Bonilla	5	Basic Switch 10/100/1000 BaseT 48 port
2019	Booth	3	Basic Switch 10/100/1000 BaseT 8 port
2020	Brewer	5	Basic Switch 10/100/1000 BaseT 48 port
2021	Bernard	7	Basic Switch 10/100/1000 BaseT 24 port

Here we use wildcard selecting all customers with Last name starting with "B" with "Basic Switch" orders and grouping them by Year_purchase.

11

```
sql_cd1.11 = "SELECT Year_Purchase,
COUNT(orders.OrderID) AS order_numbers, AVG(Cost)
FROM orders
LEFT JOIN customers ON customers.CustomerID = orders.CustomerID
WHERE City = 'Washington' OR City = 'Miami'
GROUP BY Year_Purchase;"
kable(dbGetQuery(mydb1, sql_cd1.11))
```

Year_Purchase	order_numbers	AVG(Cost)
2018	3	184.3333
2019	5	123.2000
2020	2	194.0000
2021	5	158.4000

Here we select two cities: Washington and Miami. The query gets order information and cost average in these cities each year.

```r
dbDisconnect(mydb1)
```

Part 2 HIV Database Analysis

1

```r
mydb2 <- dbConnect(RSQLite::SQLite(), "my-db2.sqlite")
str(dbGetQuery(mydb2, 'SELECT * FROM hiv_info;'))
```

```
## 'data.frame':    44 obs. of  11 variables:
##  $ Entity                           : chr  "Bermuda" "Bermuda" "Bermuda" "Bermuda" ...
##  $ Year                             : int  2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 ...
##  $ number_of_people_living_with_HIV  : int  92076 86263 81348 78050 75830 73765 71905 70292 68974 679
##  $ deaths_less_than5                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ deaths_more_than70               : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ deaths_5to14                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ deaths_15to49                    : int  5 5 5 4 4 4 4 5 5 4 ...
##  $ deaths_50to69                    : int  3 2 2 2 2 2 2 3 3 3 ...
##  $ deaths_total                     : int  8 8 7 7 7 7 7 8 8 7 ...
##  $ new_cases_of_hiv_infection       : int  5 6 6 7 7 7 7 6 6 6 ...
##  $ number_of_people_infected_with_hiv: int  92 86 81 78 76 74 72 70 69 68 ...
```

```r
str(dbGetQuery(mydb2, 'SELECT * FROM country_info;'))
```

```
## Warning in result_fetch(res@ptr, n = n): Column `GDP_per_capital`: mixed type,
## first seen values of type real, coercing other values of type string

## Warning in result_fetch(res@ptr, n = n): Column `unemployment_rate`: mixed type,
## first seen values of type string, coercing other values of type real

## Warning in result_fetch(res@ptr, n = n): Column `School_enrollment_rate`: mixed
## type, first seen values of type string, coercing other values of type real

## 'data.frame':    44 obs. of  6 variables:
##  $ Country               : chr  "Bermuda" "Bermuda" "Bermuda" "Bermuda" ...
##  $ Year                  : int  2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 ...
##  $ Country_Population    : int  64888 65273 65636 65124 64564 64798 65001 65139 65239 64555 ...
##  $ GDP_per_capital       : num  90850 93606 88463 88207 85973 ...
##  $ unemployment_rate     : chr  "Null" "Null" "Null" "Null" ...
##  $ School_enrollment_rate: chr  "Null" "Null" "Null" "Null" ...
```

There are 44 observations and 11 variables in the HIV_info table and 44 observations , 6 variables in the country_info table. From these two tables we can see that we need to combine country/Entity and year as the primary keys in order to join them.

2

```
sql_cd2.2 = "SELECT Country, hiv_info.Year, new_cases_of_hiv_infection,
School_enrollment_rate
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.Year = country_info.Year
WHERE Country = 'North America';"
df_onlyNA <- dbGetQuery(mydb2,sql_cd2.2)
# Check structure
str(df_onlyNA)
```

```
## 'data.frame':    11 obs. of  4 variables:
##  $ Country                   : chr  "North America" "North America" "North America" "North America"
##  $ Year                      : int  2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 ...
##  $ new_cases_of_hiv_infection: int  44440 43004 41891 41617 42212 43242 44717 46636 48933 51661 ...
##  $ School_enrollment_rate    : num  96.2 96.3 95 93.8 92.9 ...
```

```
# Check summary statistics
summary(df_onlyNA)
```

```
##    Country               Year      new_cases_of_hiv_infection
##  Length:11          Min.   :2007   Min.   :41617
##  Class :character   1st Qu.:2010   1st Qu.:42608
##  Mode  :character   Median :2012   Median :44440
##                     Mean   :2012   Mean   :45742
##                     3rd Qu.:2014   3rd Qu.:47784
##                     Max.   :2017   Max.   :54809
##  School_enrollment_rate
##  Min.   :92.90
##  1st Qu.:93.55
##  Median :94.37
##  Mean   :94.51
##  3rd Qu.:95.29
##  Max.   :96.26
```

```
kable(df_onlyNA)
```

Country	Year	new_cases_of_hiv_infection	School_enrollment_rate
North America	2007	44440	96.20578
North America	2008	43004	96.26419
North America	2009	41891	95.04202
North America	2010	41617	93.79489
North America	2011	42212	92.90273
North America	2012	43242	93.33829
North America	2013	44717	93.35277

Country	Year	new_cases_of_hiv_infection	School_enrollment_rate
North America	2014	46636	93.73903
North America	2015	48933	94.37224
North America	2016	51661	95.48883
North America	2017	54809	95.08227

For this query, we joined the two table with `country_info ON Entity = Country AND hiv_info.Year = country_info.Year` to find out the year, new cases of hiv infection and school enrollment rate of North America.

3

The column "number_of_people_living_with_HIV" had a problem that the values were more than the country population so we need to discard this column. Therefore we choose "number_of_people_infected_with_hiv" minus "deaths_total" to calculate people living with hiv. Also here we need to cast the values as float numbers to get decimal outputs. To calculate the death rate of hiv, we use the deaths_total divided by all people infected.

```
sql_cd2.3 = "SELECT Country, hiv_info.Year, Country_Population,
CAST((number_of_people_infected_with_hiv - deaths_total) AS FLOAT)*100
/CAST(Country_Population AS FLOAT) AS 'percentage_of_people_living_with_HIV',
CAST(deaths_total AS FLOAT)*100 /CAST(number_of_people_infected_with_hiv AS FLOAT)
AS 'percentage_deaths_from_HIV'
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.Year = country_info.Year
WHERE Country = 'North America';"
df_onlyNApercentage <- dbGetQuery(mydb2,sql_cd2.3)
kable(df_onlyNApercentage)
```

Country	Year	Country_Population	percentage_of_people_living_with_HIV	percentage_deaths_from_HIV
North America	2007	334185120	0.3819545	0.9842350
North America	2008	337406357	0.3853274	0.8845624
North America	2009	340466060	0.3891404	0.8013663
North America	2010	343396098	0.3933705	0.6925254
North America	2011	345983901	0.3981951	0.6257366
North America	2012	348653238	0.4028082	0.5792918
North America	2013	351205682	0.4078251	0.5441824
North America	2014	353888995	0.4134084	0.5169287
North America	2015	356510820	0.4197233	0.4862104
North America	2016	359245384	0.4262713	0.4999175

Country	Year	Country_Population	percentage_of_people_living_with_HIV	percentage_deaths_from_HIV
North America	2017	361751263	0.4330050	0.4839840

By using the `Cast()` clause we got the float result of percentage of people living with hiv and percentage of deaths. We calculate the percentage of people living with hiv by: `(number_of_people_infected_with_hiv - deaths_total)*100/Country_Population`.

We get the death percentage by:`deaths_total*100/number_of_people_infected_with_hiv`.

Still, we only want data from North America this time. The table we got shows that from 2007 to 2017, the percentage of people living with HIV increased from 0.382% to 0.433% while the percentage of deaths decreased from 0.984% to 0.484%. This is reasonable because more therapies can help reduce deaths while more people are getting threatened by HIV.

4

By observing to the `number_of_people_living_with_HIV` we know that this is a problematic column and we might want to take some extra steps to tidy this column before we use it. If we remove the last 3 digits of each number in `number_of_people_living_with_HIV` then it will have the same number as `number_of_people_infected_with_hiv`, thus these two columns actually refer to same group of people and if we want to use `number_of_people_living_with_HIV` in our query then we need to divide it by 1000 here:

```
sql_cd2.4="SELECT Country ,SUM(Country_Population)
AS Sum_country_population,
CAST(SUM(number_of_people_living_with_HIV/1000*100) AS FLOAT)/CAST(SUM(Country_Population) AS FLOAT)
AS 'HIV_patient_percentage'
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.Year = country_info.Year
WHERE Country != 'North America' GROUP BY Country;"
df_2.4 <-dbGetQuery(mydb2,sql_cd2.4)
kable(df_2.4)
```

Country	Sum_country_population	HIV_patient_percentage
Bermuda	714091	0.1159516
Canada	381696529	0.2098274
United States	3450282298	0.4295326

We calculated the total population of a country from 2007-2017 and its total HIV patient percentage in this period by using `number_of_people_living_with_HIV/1000` to divide the total population to get the HIV patient percentage of Bermuda, Canada and United States. And the `group by` clause here is to help us to get the `HIV_patient_percentage` of these 3 countries individually. The US has the highest HIV patient percentage.

5

Query 1 This query helps to investigate relationship between infection rate and unemployment rate in Canada. Here we use new cases divided by healthy population (`Country_population minus number_of_people_infected_with_hiv`) to get infection rate. We select data with School_enrollment_rate >99.5 to minimise the effect of school enrollment.

```r
sql_cd2.5.1 = "SELECT Country, hiv_info.Year,
CAST(new_cases_of_hiv_infection AS FLOAT)/
CAST(Country_population-number_of_people_infected_with_hiv AS FLOAT)
AS infection_rate, unemployment_rate
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.year = country_info.year
WHERE Country = 'Canada' AND School_enrollment_rate >99.5;"
df_5.1 <- dbGetQuery(mydb2, sql_cd2.5.1)
cor(log(df_5.1$infection_rate),df_5.1$unemployment_rate)
```

```
## [1] -0.4235511
```

```r
model_5.1 <- lm(log(infection_rate)~unemployment_rate,data=df_5.1)
summary(model_5.1)
```

```
##
## Call:
## lm(formula = log(infection_rate) ~ unemployment_rate, data = df_5.1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19977 -0.10540 -0.05207  0.13774  0.23228
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -8.91362    0.54725 -16.288 3.41e-06 ***
## unemployment_rate -0.08843    0.07721  -1.145    0.296
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1775 on 6 degrees of freedom
## Multiple R-squared:  0.1794, Adjusted R-squared:  0.04263
## F-statistic: 1.312 on 1 and 6 DF,  p-value: 0.2957
```

```r
kable(df_5.1)
```

Country	Year	infection_rate	unemployment_rate
Canada	2007	6.70e-05	6.036
Canada	2008	6.40e-05	6.137
Canada	2009	6.19e-05	8.344
Canada	2010	6.18e-05	8.056
Canada	2011	6.35e-05	7.511
Canada	2015	8.20e-05	6.906
Canada	2016	8.90e-05	6.999
Canada	2017	9.69e-05	6.340

We can find they are negatively correlated. If unemployment increases, the infection rate will drop.If there is more people without work, the infection of HIV will go down. The p-value 0.296 shows that the correlation is not significant.

```
sql_cd2.5.2 ="SELECT Country, hiv_info.Year,
CAST(deaths_15to49 AS FLOAT)/CAST(deaths_total AS FLOAT)
AS death_15to49_percentage, CAST(deaths_50to69 AS FLOAT)/CAST(deaths_total AS FLOAT)
AS death_50to69_percentage
FROM hiv_info INNER JOIN country_info ON Entity = Country AND hiv_info.year = country_info.year
WHERE Country = 'United States' AND hiv_info.Year BETWEEN 2011 AND 2015;"
df_5.2 <- dbGetQuery(mydb2, sql_cd2.5.2)
kable(df_5.2)
```

Query 2

Country	Year	death_15to49_percentage	death_50to69_percentage
United States	2011	0.5173939	0.4479378
United States	2012	0.4922765	0.4712585
United States	2013	0.4753192	0.4878241
United States	2014	0.4592896	0.5002732
United States	2015	0.4445867	0.5127330

This query suggests that the percentage of deaths between 50 to 69 age is increasing from 2011 to 2015, which means the survival time of people infected with HIV could be longer as medical conditions gets better.

Query 3 Query number 3 we want to calculate the decreasing case of HIV by each year in Bermuda.

```
sql_cd2.5.3 <-"SELECT hiv_info.Year,number_of_people_infected_with_hiv
AS this_year_HIV_total,
number_of_people_infected_with_hiv - LAG (number_of_people_infected_with_hiv)
OVER (ORDER BY hiv_info.Year ASC)
AS HIV_decrease_cases,
LEAD (number_of_people_infected_with_hiv, 1) OVER (ORDER BY hiv_info.Year ASC)
AS next_year_HIV_total
FROM hiv_info INNER JOIN country_info ON Entity = Country AND hiv_info.year = country_info.year
WHERE Country = 'Bermuda';"
df_5.3<-dbGetQuery(mydb2, sql_cd2.5.3)
# check result:
kable(df_5.3)
```

Year	this_year_HIV_total	HIV_decrease_cases	next_year_HIV_total
2007	92	NA	86
2008	86	-6	81
2009	81	-5	78
2010	78	-3	76
2011	76	-2	74
2012	74	-2	72
2013	72	-2	70
2014	70	-2	69
2015	69	-1	68
2016	68	-1	67

Year	this_year_HIV_total	HIV_decrease_cases	next_year_HIV_total
2017	67	-1	NA

In this query we used the `LAG` and `over` clause to find out that Bermuda is keeping a decreasing trend each year on total HIV infected patient.

Query 4 Query number 4, in this query we want to know if the HIV death rate is the same year with the highest unemployment rate and we use the death rate of HIV per 1000000 people on different years in Canada.

```
sql_cd2.5.4 <-"SELECT hiv_info.Year, Country, Country_Population,
CAST(deaths_total * 1000000 AS FLOAT)/CAST(Country_Population AS FLOAT)
AS 'deaths_rate',GDP_per_capital,unemployment_rate, School_enrollment_rate
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.year = country_info.year
WHERE Country = 'Canada'
ORDER BY (CASE WHEN School_enrollment_rate IS NULL
THEN unemployment_rate
ELSE unemployment_rate
END);"
df_5.4<-dbGetQuery(mydb2, sql_cd2.5.4)
```

```
## Warning in result_fetch(res@ptr, n = n): Column `School_enrollment_rate`: mixed
## type, first seen values of type string, coercing other values of type real
```

```
# check result:
kable(df_5.4)
```

Year	Country	Country_Population	deaths_rate	GDP_per_capital	unemployment_rate	School_enrollment_rate
2007	Canada	32889025	13.651970	44543.04	6.036	Null
2008	Canada	33247118	12.572518	46594.45	6.137	Null
2017	Canada	36540268	7.498577	45069.93	6.340	99.8825
2015	Canada	35702908	7.842498	43495.05	6.906	99.58907
2014	Canada	35437435	7.901249	50835.51	6.914	99.39113
2016	Canada	36109487	7.698808	42279.90	6.999	99.95587
2013	Canada	35082954	6.755418	52504.66	7.074	99.43231
2012	Canada	34714222	8.152278	52542.35	7.292	99.45167
2011	Canada	34339328	8.969308	52101.80	7.511	Null
2010	Canada	34004889	9.851525	47450.32	8.056	Null
2009	Canada	33628895	10.913234	40773.06	8.344	Null

We want to order the output table by `School_enrollment_rate` but if it is null then we will alternatively order by `unemployment_rate`(and since we knew there are null value in school enrollment rate so we know it will order by unemployment rate) .As we can see that there is some null value in the `School_enrollment_rate` so it is ordered by `unemployment_rate` instead with the help of `CASE WHEN` clause. The highest death rate was on 2007 but we actually have the lowest unemployment rate this year, so the HIV death rate might have no huge impact on the unemployment rate since they are not in the same year.

Query 5. In this query we want to find out the rank of `new_cases_of_hiv_infection` each year and partition it by country:

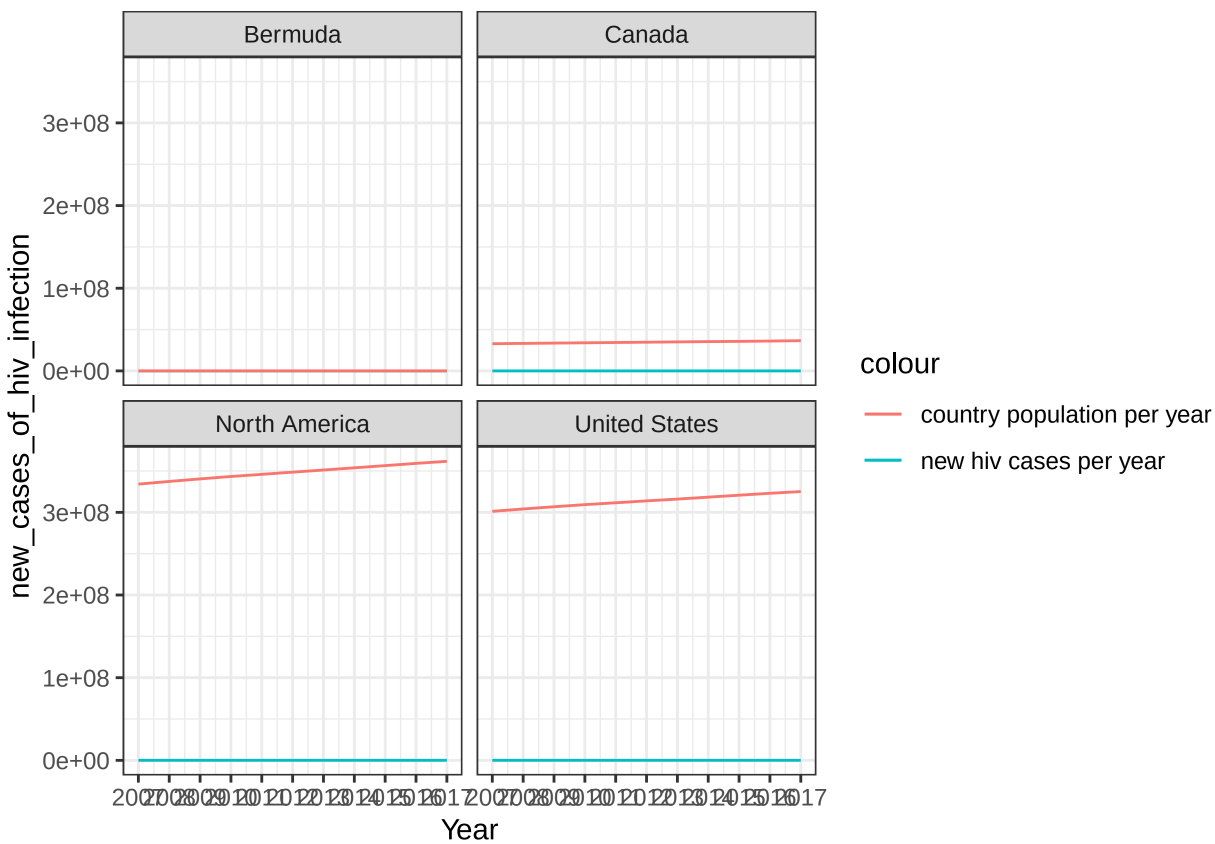
```
sql_cd2.5.5 <-"select Country, hiv_info.Year,Country_Population,new_cases_of_hiv_infection ,
dense_rank() over (partition by Country order by new_cases_of_hiv_infection DESC)
AS 'rank'
FROM hiv_info INNER JOIN country_info ON Entity = Country
AND hiv_info.year = country_info.year;"
df_5.5<-dbGetQuery(mydb2, sql_cd2.5.5)
# check result:
kable(df_5.5)
```

Country	Year	Country_Population	new_cases_of_hiv_infection	rank
Bermuda	2010	65124	7	1
Bermuda	2011	64564	7	1
Bermuda	2012	64798	7	1
Bermuda	2013	65001	7	1
Bermuda	2008	65273	6	2
Bermuda	2009	65636	6	2
Bermuda	2014	65139	6	2
Bermuda	2015	65239	6	2
Bermuda	2016	64555	6	2
Bermuda	2017	63874	6	2
Bermuda	2007	64888	5	3
Canada	2017	36540268	3532	1
Canada	2016	36109487	3206	2
Canada	2015	35702908	2920	3
Canada	2014	35437435	2674	4
Canada	2013	35082954	2467	5
Canada	2012	34714222	2301	6
Canada	2007	32889025	2201	7
Canada	2011	34339328	2177	8
Canada	2008	33247118	2125	9
Canada	2010	34004889	2096	10
Canada	2009	33628895	2079	11
North America	2017	361751263	54809	1
North America	2016	359245384	51661	2
North America	2015	356510820	48933	3
North America	2014	353888995	46636	4
North America	2013	351205682	44717	5
North America	2007	334185120	44440	6
North America	2012	348653238	43242	7
North America	2008	337406357	43004	8
North America	2011	345983901	42212	9
North America	2009	340466060	41891	10
North America	2010	343396098	41617	11
United States	2017	325147121	51266	1
United States	2016	323071342	48445	2
United States	2015	320742673	46003	3
United States	2014	318386421	43952	4
United States	2013	316057727	42240	5
United States	2007	301231207	42232	6

Country	Year	Country_Population	new_cases_of_hiv_infection	rank
United States	2012	313874218	40931	7
United States	2008	304093966	40871	8
United States	2011	311580009	40026	9
United States	2009	306771529	39804	10
United States	2010	309326085	39513	11

The `partition by` helps us to get a rank within same country but different years and `dense_rank()` clause is a window function, which ranking in descending ordered partition of `new_cases_of_hiv_infection`. From the result we can see that the Bermuda has the most `new_cases_of_hiv_infection` on 2010, 2011, 2012, 2013; Canada reached peak at 2017; North America reaches peak at 2017; and US has the most new cases on 2017. However, I think the new cases of HIV is increasing accordingly with the country population and we can actually use ggplot2 to visualization our result.

```
ggplot(df_5.5, aes(x = Year, y = new_cases_of_hiv_infection,
                   color="new hiv cases per year" )) +
    geom_line()+
  geom_line(aes(y=Country_Population, color="country population per year")) +
  scale_x_continuous(breaks=seq(2007, 2017, 1))+
  facet_wrap(. ~ Country)+
  theme_bw()
```



So, we can see that basically for every countries but Bermuda have a increasing trending of total population and their HIV new cases are relatively have no change in trend at all. Bermuda have a overlapped result but given the fact that its total population is not huge and each year of new HIV cases has no more than 10 we still can say that all of these countries have a ideal HIV new cases each year. The HIV disease in these countries is under control.

```
dbDisconnect(mydb2)
```