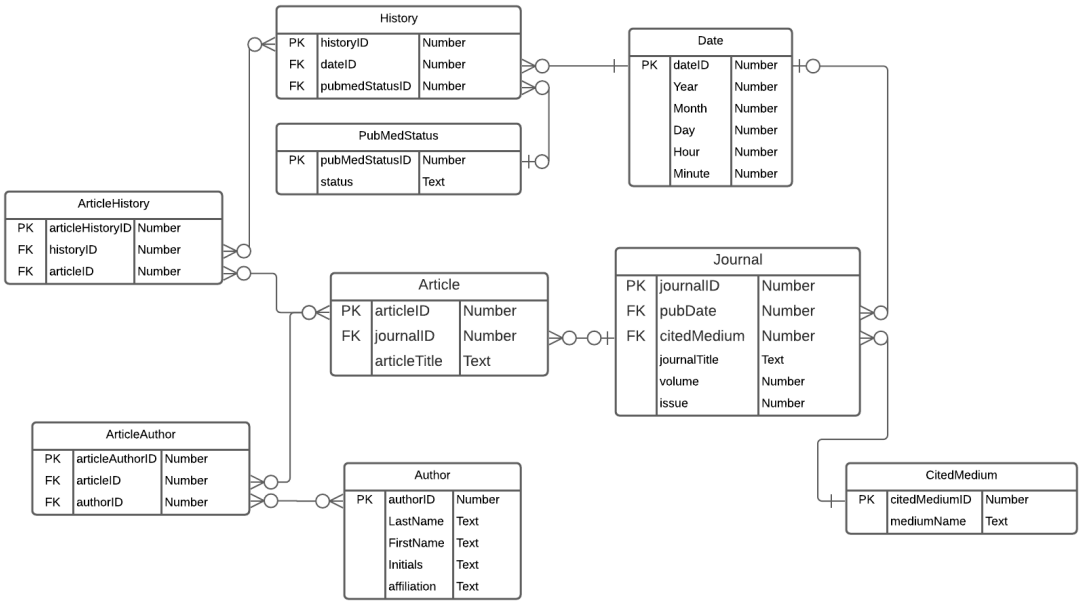
Code

Datamining a Medical Journal

Part 1.1 Create relational schema



Diagram

Part 2.2 Realize schema in SQLite

```
library(RSQLite)

fpath = "~/Documents/cs5200/sqlite/"
dbfile = "practicumII"
```

```
dbcon <- dbConnect(RSQLite::SQLite(),
paste0(fpath,dbfile))
```

Create CitedMedium table

```
DROP TABLE IF EXISTS CitedMedium
CREATE TABLE CitedMedium (
  citedMediumID INTEGER PRIMARY KEY,
  mediumName TEXT NOT NULL UNIQUE
)
```

Create Affiliation table

```
DROP TABLE IF EXISTS Affiliation
CREATE TABLE Affiliation (
  affiliationID INTEGER PRIMARY KEY,
  affiliationName TEXT NOT NULL UNIQUE
)
```

Create PubMedStatus table

```
DROP TABLE IF EXISTS PubMedStatus
CREATE TABLE PubMedStatus (
  pubMedStatusID INTEGER PRIMARY KEY,
  status TEXT NOT NULL UNIQUE
)
```

Create Date table

```
DROP TABLE IF EXISTS Date
CREATE TABLE Date (
  dateID INTEGER PRIMARY KEY,
  year INTEGER DEFAULT NULL,
  month INTEGER DEFAULT NULL,
  day INTEGER DEFAULT NULL,
  hour INTEGER DEFAULT NULL,
  minute INTEGER DEFAULT NULL
)
```

Create History table

```
DROP TABLE IF EXISTS History
CREATE TABLE History (
  historyID INTEGER PRIMARY KEY,
  dateID INTEGER NOT NULL REFERENCES Date(dateID),
  pubmedStatusID INTEGER REFERENCES
PubMedStatus(pubMedStatusID)
)
```

Create Author table

```
DROP TABLE IF EXISTS Author
CREATE TABLE Author (
  authorID INTEGER PRIMARY KEY,
  lastName TEXT,
  firstName TEXT,
  initials TEXT,
  affiliation TEXT
)
```

Create Journal table

```
DROP TABLE IF EXISTS Journal
CREATE TABLE Journal (
  journalID INTEGER PRIMARY KEY,
  pubDateID INTEGER REFERENCES Date(dateID),
  citedMedium INTEGER REFERENCES
CitedMedium(citedMediumID),
  journalTitle TEXT,
  volume INTEGER DEFAULT NULL,
  issue INTEGER DEFAULT NULL
)
```

Create Article table

```
DROP TABLE IF EXISTS Article
CREATE TABLE Article (
```

```
  articleID INTEGER PRIMARY KEY,
  journalID INTEGER NOT NULL REFERENCES
Article(articleID),
  articleTitle TEXT
)
```

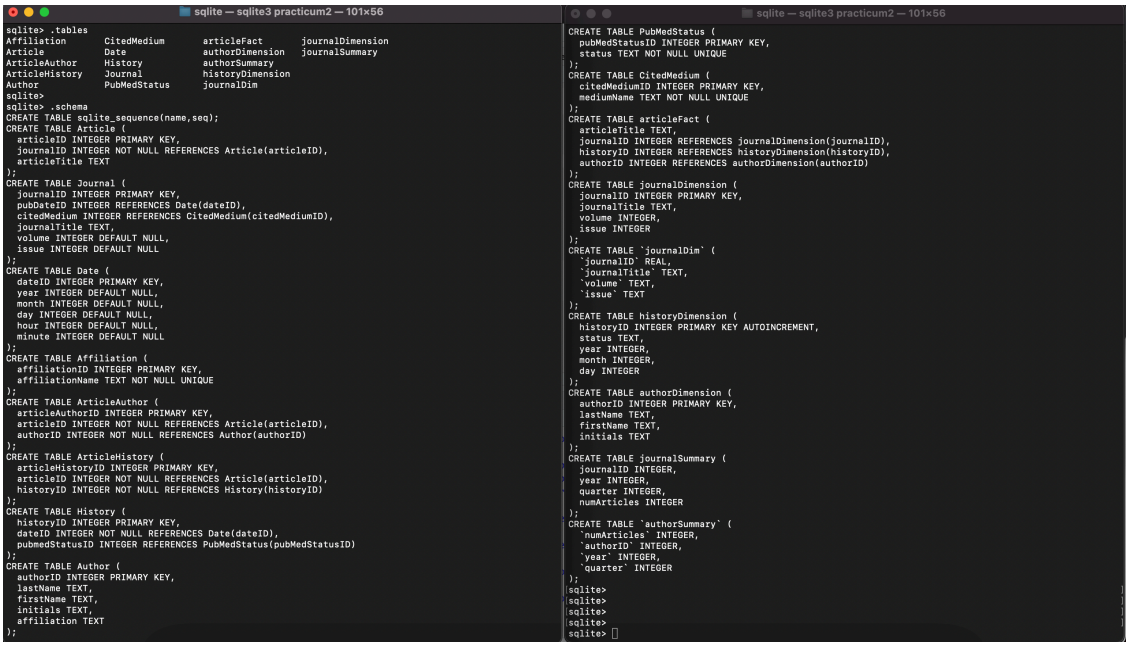
Create ArticleAuthor table

```
DROP TABLE IF EXISTS ArticleAuthor
CREATE TABLE ArticleAuthor (
  articleAuthorID INTEGER PRIMARY KEY,
  articleID INTEGER NOT NULL REFERENCES
Article(articleID),
  authorID INTEGER NOT NULL REFERENCES Author(authorID)
)
```

Create ArticleHistory table

```
DROP TABLE IF EXISTS ArticleHistory
CREATE TABLE ArticleHistory (
  articleHistoryID INTEGER PRIMARY KEY,
  articleID INTEGER NOT NULL REFERENCES
Article(articleID),
  historyID INTEGER NOT NULL REFERENCES
History(historyID)
)
```

Inspected db



Diagram

Part 1.3 Import XML and populate tables

Import xml

```
library(XML)
path <- "/Users/jdenman/Downloads/"
xmlFile <- "pubmed_sample.xml"
fp <- paste0(path,xmlFile)

xmlObj <- xmlParse(fp)
xmlObjTree <- xmlTreeParse(fp)

r <- xmlRoot(xmlObj)
```

Create dataframes

```
numArticles <- xmlSize(r)

Article.df <- data.frame (articleID = vector (mode =
```

```
"integer",
                                                  length =
numArticles),
                    articleTitle = vector (mode =
"character",
                                           length =
numArticles),
                     journalID = vector (mode =
"integer",
                                        length =
numArticles),
                     stringsAsFactors = F)


Journal.df <- data.frame (journalID = integer(),
                          pubDateID = integer(),
                          journalTitle = character(),
                          volume = integer(),
                          issue = integer(),
                          citedMedium = integer(),
                          stringsAsFactors = F)

Author.df <- data.frame (authorID = integer(),
                         lastName = character(),
                         firstName = character(),
                         initials = character(),
                         affiliation = character(),
                           stringsAsFactors = F)

History.df <- data.frame (historyID = integer(),
                         dateID = integer(),
                         pubmedStatusID = integer(),
                           stringsAsFactors = F)

Date.df <- data.frame (dateID = integer(),
```

```
                          year = integer(),
                         month = integer(),
                         day = integer(),
                         hour = integer(),
                         minute = integer(),
                            stringsAsFactors = F)

PubMedStatus.df <- data.frame (pubMedStatusID =
integer(),
                          status = character(),
                            stringsAsFactors = F)

CitedMedium.df <- data.frame (citedMediumID =
integer(),
                         mediumName = character(),
                            stringsAsFactors = F)

Affiliation.df <- data.frame (affiliationID =
integer(),
                         affiliation = character(),
                            stringsAsFactors = F)

ArticleAuthor.df <- data.frame (articleAuthorID =
integer(),
                         articleID = integer(),
                         authorID = integer(),
                            stringsAsFactors = F)

ArticleHistory.df <- data.frame (articleHistoryID =
integer(),
                         articleID = integer(),
                         historyID = integer(),
                            stringsAsFactors = F)
```

helper function from course material example

```
rowExists <- function (aRow, aDF)
{
  # check if that address is already in the data frame
  n <- nrow(aDF)
  c <- ncol(aDF)
  if (n == 0)
  {
    # data frame is empty, so can't exist
    return(0)
  }

  for (a in 1:n)
  {
    # check if all columns match for a row; ignore the
aID column
    if (all(aDF[a,] == aRow[1,]))
    {
      # found a match; return it's ID
      return(a)
    }
  }

  # none matched
  return(0)
}
parseJournals <- function (aJournalsNode)
{
  newJournal.df <- data.frame(
                               pubDate = integer(),
                               citedMedium =
character(),
                               journalTitle =
character(),
                             volume = integer(),
                             issue = integer(),
```

```
                              citedMediumID = integer(),
                              year = integer(),
                              month = integer(),
                              day = integer(),
                              stringsAsFactors = F)
  n <- xmlSize(aJournalsNode)

  # extract each of the <Item> nodes under <Items>
  for (m in 1:n)
  {
    aJournal <- aJournalsNode[[m]]

    title <- xpathSApply(aJournal, "./Title", xmlValue)
    if (length(title) == 0)
      title <- ""

    citedMedium <- xpathSApply(aJournal, "./
JournalIssue/@CitedMedium")
    if (length(citedMedium) == 0)
      citedMedium <- ""

    year <- xpathSApply(aJournal, "./JournalIssue/
PubDate/Year", xmlValue)
    if (length(year) == 0)
      year <- 0

    month <- xpathSApply(aJournal, "./JournalIssue/
PubDate/Month", xmlValue)
    month <- match(month, month.abb)

    if (length(month) == 0)
      month <- 0

    day <- xpathSApply(aJournal, "./JournalIssue/
PubDate/Day", xmlValue)
```

```
    if (length(day) == 0)
      day <- 0

    volume <- xpathSApply(aJournal, "./JournalIssue/
Volume", xmlValue)
    if (length(volume) == 0)
      volume <- 0

    issue <- xpathSApply(aJournal, "./JournalIssue/
Issue", xmlValue)
    if (length(issue) == 0)
      issue <- 0

    newJournal.df[m,1] <- 0
    newJournal.df[m,2] <- citedMedium
    newJournal.df[m,3] <- title
    newJournal.df[m,4] <- volume
    newJournal.df[m,5] <- issue
    newJournal.df[m,6] <- citedMediumID
    newJournal.df[m,7] <- year
    newJournal.df[m,8] <- month
    newJournal.df[m,9] <- day
  }

  return(newJournal.df)
}
```

parse xml and add to dataframes

```
for (i in 1:numArticles)
{
  anArticle <- r[[i]]

  Article.df$articleID[i] <- i

  # add article title to article df
```

```
  xpath <- paste0("//Article/ArticleTitle")
  title <- xpathSApply(anArticle, xpath, xmlValue)
  Article.df$articleTitle[i] <- title[i]

  # get journal node
  xpath <- paste0("./MedlineCitation/Article/Journal")
  journalNode <- xpathSApply(anArticle, xpath)
  xmlSize(journalNode)

  articleJournal <- parseJournals(journalNode)
  articleJournal[1,]



  citedMedium <- articleJournal$citedMedium

  newCitedMed.df <- data.frame(citedMedium =
citedMedium, stringsAsFactors = F)

  nrow(CitedMedium.df[,2])
    citedMediumID <- rowExists(newCitedMed.df,
CitedMedium.df[,2, drop=FALSE])

    if (citedMediumID == 0) {
      citedMediumID <- nrow(CitedMedium.df) + 1

CitedMedium.df[citedMediumID,2:ncol(CitedMedium.df)] <-
citedMedium #shipping[1,]
      CitedMedium.df[citedMediumID,1] <- citedMediumID
    }

    articleJournal$citedMediumID <- citedMediumID

    newDate.df <- data.frame (year = vector (mode =
"integer",
                                             length =
```

```
1),
                    month = vector (mode = "integer",
                                    length = 1),
                     day = vector (mode = "integer",
                                   length = 1),
                    hour = vector (mode = "integer",
                                   length = 1),
                    minute = vector (mode = "integer",
                                   length = 1),
                     stringsAsFactors = F)

    newDate.df$year <- articleJournal$year
    newDate.df$month <- articleJournal$month
    newDate.df$day <- articleJournal$day
    newDate.df$hour <- 0
    newDate.df$minute <- 0

    dateID <- rowExists(newDate.df,
Date.df[,2:ncol(Date.df)])
    if (dateID == 0) {
      dateID <- nrow(Date.df) + 1
      Date.df[dateID,2:ncol(Date.df)] <- newDate.df
      Date.df[dateID,1] <- dateID
    }

    articleJournal$pubDate <- dateID

  # drop columns that aren't in Journal.df
  articleJournal$citedMedium <- NULL
  articleJournal$year <- NULL
  articleJournal$month <- NULL
  articleJournal$day <- NULL

  journalID <- rowExists(articleJournal,
Journal.df[,2:ncol(Journal.df)])
```

```
  if (journalID == 0)
    {
      journalID <- nrow(Journal.df) + 1
      Journal.df[journalID,2:ncol(Journal.df)] <-
articleJournal #shipping[1,]
      Journal.df[journalID,1] <- journalID
    }

  Article.df$journalID[i] <- journalID

  #get authors
  xpath <- paste0("./MedlineCitation/Article/
AuthorList//Author")
  authorList <- xpathSApply(anArticle, xpath)
  numAuthors <- xmlSize(authorList)
  for (m in 1:numAuthors) {
    newAuthor.df <- data.frame(lastname = character(),
forname = character(),
                               initials = character(),
affiliation = character(), stringsAsFactors = FALSE)
    anAuthor <- authorList[[m]]

    lastname <- xpathSApply(anAuthor, "./LastName",
xmlValue)
    if (length(lastname) == 0)
      lastname <- ""

    firstname <- xpathSApply(anAuthor, "./ForeName",
xmlValue)
    if (length(firstname) == 0)
      firstname <- ""

    initials <- xpathSApply(anAuthor, "./Initials",
xmlValue)
```

```
    if (length(initials) == 0)
      initials <- ""

    affiliation <- xpathSApply(anAuthor, "./
Affiliation", xmlValue)
    if (length(affiliation) == 0)
      affiliation <- ""

    newAuthor.df[1,1] <- lastname
    newAuthor.df[1,2] <- firstname
    newAuthor.df[1,3] <- initials
    newAuthor.df[1,4] <- affiliation

    authorID <- rowExists(newAuthor.df,
Author.df[,2:ncol(Author.df)])

    if (authorID == 0) {
      authorID <- nrow(Author.df) + 1
      Author.df[authorID,2:ncol(Author.df)] <-
newAuthor.df #shipping[1,]
      Author.df[authorID,1] <- authorID
    }
    newArticleAuthor.df <- data.frame(articleID = i,
authorID = authorID )
    articleAuthorID <- rowExists(newArticleAuthor.df,
ArticleAuthor.df[,2:ncol(ArticleAuthor.df)])
    if (articleAuthorID == 0) {
      articleAuthorID <- nrow(ArticleAuthor.df) + 1

ArticleAuthor.df[articleAuthorID,2:ncol(ArticleAuthor.d
f)] <- newArticleAuthor.df #shipping[1,]
      ArticleAuthor.df[articleAuthorID,1] <-
articleAuthorID
    }
```

```
  }

  #get history
  xpath <- paste0("./PubmedData/History//
PubMedPubDate")
  historyList <- xpathSApply(anArticle, xpath)
  numHistory <- xmlSize(historyList)
  for (n in 1:numHistory) {
    newHistory <- data.frame(dateID = integer(),
pubmedStatusID = integer(), stringsAsFactors = FALSE)

    aHistory <- historyList[[n]]

    status <- xpathSApply(aHistory, "./@PubStatus")
    if (length(status) == 0)
      status <- ""

    year <- xpathSApply(aHistory, "./Year", xmlValue)
    if (length(year) == 0)
      year <- 0

    month <- xpathSApply(aHistory, "./Month", xmlValue)
    if (length(month) == 0)
      month <- 0

    day <- xpathSApply(aHistory, "./Day", xmlValue)
    if (length(day) == 0)
      day <- 0

    hour <- xpathSApply(aHistory, "./Hour", xmlValue)
    if (length(hour) == 0)
      hour <- 0

    minute <- xpathSApply(aHistory, "./Minute",
xmlValue)
```

```
    if (length(minute) == 0)
      minute <- 0

    newpubstatus.df <- data.frame(status = status,
stringsAsFactors = FALSE)
    pubstatusID <- rowExists(newpubstatus.df,
PubMedStatus.df[,2, drop=FALSE])

    if (pubstatusID == 0) {
      pubstatusID <- nrow(PubMedStatus.df) + 1

PubMedStatus.df[pubstatusID,2:ncol(PubMedStatus.df)] <-
newpubstatus.df #shipping[1,]
      PubMedStatus.df[pubstatusID,1] <- pubstatusID
    }

    newdate.df <- data.frame(year = year, month =
month, day = day, hour = hour, minute = minute)
    dateID <- rowExists(newdate.df,
Date.df[,2:ncol(Date.df)])

    if (dateID == 0) {
      dateID <- nrow(Date.df) + 1
      Date.df[dateID,2:ncol(Date.df)] <- newdate.df
#shipping[1,]
      Date.df[dateID,1] <- dateID
    }

    newHistory.df <- data.frame(dateID = dateID,
pubmedStatusID = pubstatusID)
    historyID <- rowExists(newHistory.df,
History.df[,2:ncol(History.df)])

    if (historyID == 0) {
      historyID <- nrow(History.df) + 1
```

```
      History.df[historyID,2:ncol(History.df)] <-
newHistory.df #shipping[1,]
      History.df[historyID,1] <- historyID
    }

    newArticleHistory.df <- data.frame(articleID = i,
historyID = historyID )
    articleHistoryID <- rowExists(newArticleHistory.df,
ArticleHistory.df[,2:ncol(ArticleHistory.df)])
    if (articleHistoryID == 0) {
      articleHistoryID <- nrow(ArticleHistory.df) + 1

ArticleHistory.df[articleHistoryID,2:ncol(ArticleAuthor
.df)] <- newArticleHistory.df #shipping[1,]
      ArticleHistory.df[articleHistoryID,1] <-
articleHistoryID
    }
  }
}
```

write to Article

```
#library(RSQLite)
dbWriteTable(dbcon, "Article", Article.df, append =
TRUE, row.names = FALSE)
```

write to Journal

```
#library(RSQLite)
dbWriteTable(dbcon, "Journal", Journal.df, append =
TRUE, row.names = FALSE)
```

write to Date

```
#library(RSQLite)
dbWriteTable(dbcon, "Date", Date.df, append = TRUE,
row.names = FALSE)
```

write to History

```
#library(RSQLite)
dbWriteTable(dbcon, "History", History.df, append =
TRUE, row.names = FALSE)
```

write to Author

```
#library(RSQLite)
dbWriteTable(dbcon, "Author", Author.df, append = TRUE,
row.names = FALSE)
```

write to ArticleHistory

```
#library(RSQLite)
dbWriteTable(dbcon, "ArticleHistory",
ArticleHistory.df, append = TRUE, row.names = FALSE)
```

write to ArticleAuthor

```
#library(RSQLite)
dbWriteTable(dbcon, "ArticleAuthor", ArticleAuthor.df,
append = TRUE, row.names = FALSE)
```

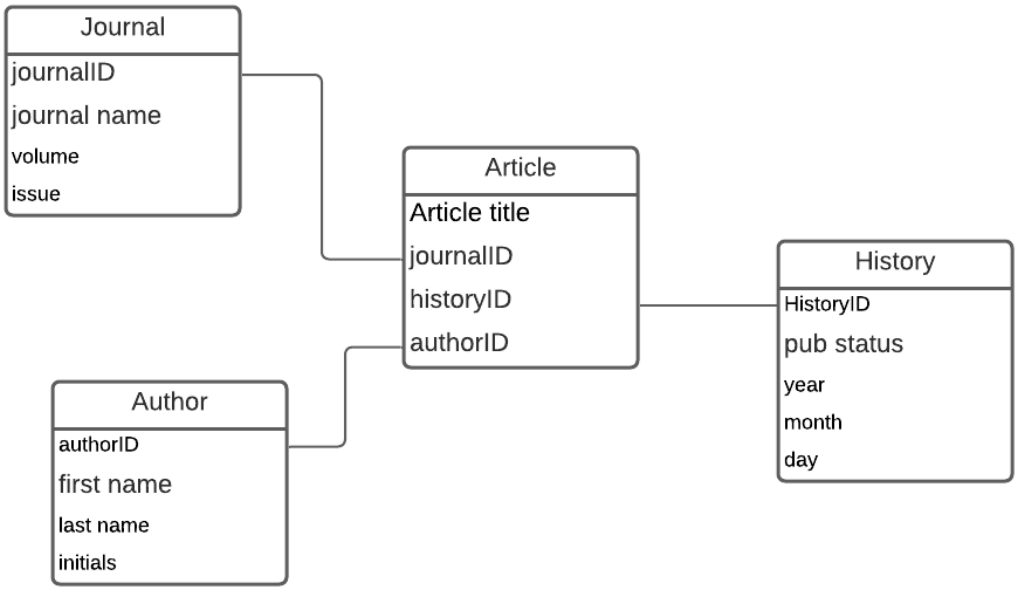
write to PubMedStatus

```
#library(RSQLite)
dbWriteTable(dbcon, "PubMedStatus", PubMedStatus.df,
append = TRUE, row.names = FALSE)
```

write to citedMedium

```
#library(RSQLite)
dbWriteTable(dbcon, "CitedMedium", CitedMedium.df,
append = TRUE, row.names = FALSE)
```

Part 2.1 Create fact tables and star schema



Diagram

Create journal dimension table

```
DROP TABLE IF EXISTS journalDimension
CREATE TABLE journalDimension (
  journalID INTEGER PRIMARY KEY,
  journalTitle TEXT,
  volume INTEGER,
  issue INTEGER
)
```

Create history dimension table

```
DROP TABLE IF EXISTS historyDimension
CREATE TABLE historyDimension (
  historyID INTEGER PRIMARY KEY AUTOINCREMENT,
  status TEXT,
  year INTEGER,
  month INTEGER,
  day INTEGER
)
```

Create author dimension table

```
DROP TABLE IF EXISTS authorDimension
CREATE TABLE authorDimension (
  authorID INTEGER PRIMARY KEY,
  lastName TEXT,
  firstName TEXT,
  initials TEXT
)
```

Create article fact table

```
DROP TABLE IF EXISTS articleFact
CREATE TABLE articleFact (
  articleTitle TEXT,
  journalID INTEGER REFERENCES
journalDimension(journalID),
  historyID INTEGER REFERENCES
historyDimension(historyID),
  authorID INTEGER REFERENCES authorDimension(authorID)
)
```

populate journal dimension

```
journalDim.df <- Journal.df

journalDim.df$pubDateID <- NULL
journalDim.df$citedMedium <- NULL
#library(RSQLite)
dbWriteTable(dbcon, "journalDimension", journalDim.df,
append = TRUE, row.names = FALSE)
```

populate history dimension

```
#History.df
library(sqldf)

sqlCmd = "SELECT PubMedStatus.status
```

```
               FROM History
               LEFT JOIN PubMedStatus
               ON History.pubmedStatusID =
PubMedStatus.pubmedStatusID
               "

# send the SQL query to the database
status = dbGetQuery(dbcon, sqlCmd)

sqlCmd = "SELECT Date.year, Date.month, Date.day
               FROM History
               LEFT JOIN Date
               ON History.dateID = Date.dateID
               "

# send the SQL query to the database
date = dbGetQuery(dbcon, sqlCmd)
date$status <- status$status
#library(RSQLite)
dbWriteTable(dbcon, "historyDimension", date, append =
TRUE, row.names = FALSE)
```

populate author dimension

```
authorDim.df <- Author.df

authorDim.df$affiliation <- NULL
#library(RSQLite)
dbWriteTable(dbcon, "authorDimension", authorDim.df,
append = TRUE, row.names = FALSE)
```

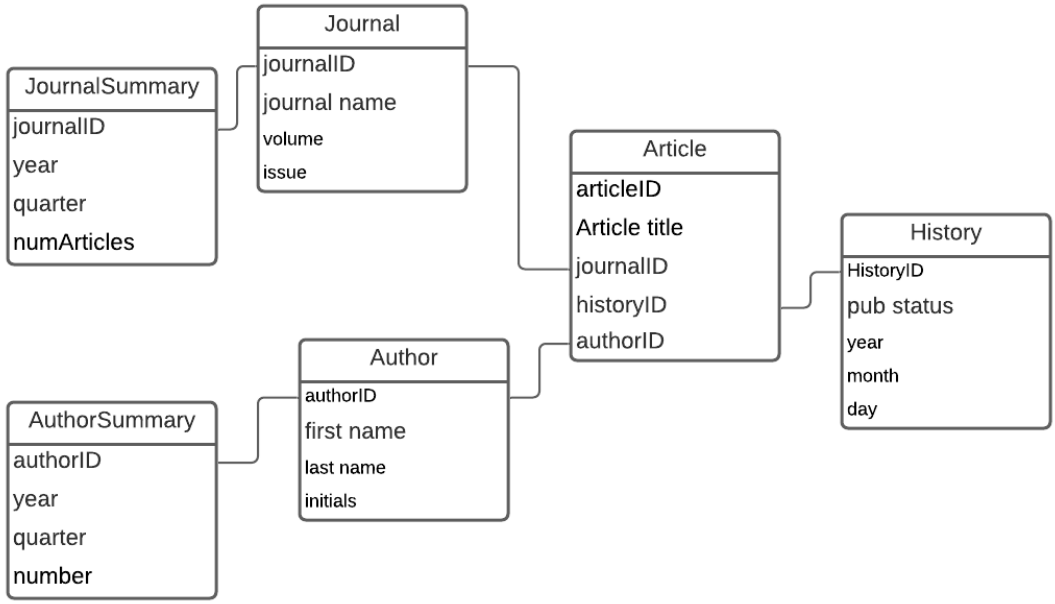
populate article fact table

```
#History.df
library(sqldf)

sqlCmd = "SELECT Article.articleTitle,
```

```
journalDimension.journalID, historyDimension.historyID,
authorDimension.authorID
FROM Article
JOIN journalDimension
ON Article.journalID = journalDimension.journalID
JOIN ArticleHistory
ON Article.articleID = ArticleHistory.articleID
JOIN historyDimension
ON ArticleHistory.historyID =
historyDimension.historyID
JOIN ArticleAuthor
ON Article.articleID = ArticleAuthor.articleID
JOIN authorDimension
ON ArticleAuthor.authorID = authorDimension.authorID
"

# send the SQL query to the database
article = dbGetQuery(dbcon, sqlCmd)
head(article)
```

```
#library(RSQLite)
dbWriteTable(dbcon, "articleFact", article, append =
TRUE, row.names = FALSE)
```

Part 2.2



Diagram

Create journal summary table

```
DROP TABLE IF EXISTS journalSummary
CREATE TABLE journalSummary (
  journalID INTEGER,
  year INTEGER,
  quarter INTEGER,
  numArticles INTEGER
)
library(sqldf)

sqlCmd = "SELECT COUNT(*) as numArticles, journalID,
year,
CASE
  WHEN month < 4 THEN 1
  WHEN month > 4 AND month < 8 THEN 2
  WHEN month > 8 AND month < 10 THEN 3
  ELSE 4
  END quarter
FROM (
```

```
SELECT journalDimension.journalID,
Article.articleTitle, historyDimension.year,
historyDimension.month
FROM journalDimension
JOIN Article
ON Article.journalID = journalDimension.journalID
JOIN articleFact
ON Article.articleTitle = articleFact.articleTitle
JOIN historyDimension
ON articleFact.historyID = historyDimension.historyID
WHERE historyDimension.status = 'pubmed'
GROUP BY journalDimension.journalID,
historyDimension.year, historyDimension.month
)
GROUP BY journalID, year, quarter

"


# send the SQL query to the database
journalSum = dbGetQuery(dbcon, sqlCmd)
head(journalSum)
```

```
#library(RSQLite)
dbWriteTable(dbcon, "journalSummary", journalSum,
append = TRUE, row.names = FALSE)
```

Create author summary table

```
DROP TABLE IF EXISTS authorSummary
CREATE TABLE authorSummary (
  authorID INTEGER,
  year INTEGER,
  quarter INTEGER,
  numArticles INTEGER
)
library(sqldf)
```

```
sqlCmd = "SELECT COUNT(*) as  numArticles, authorID,
year,
CASE
  WHEN month < 4 THEN 1
  WHEN month > 4 AND month < 8 THEN 2
  WHEN month > 8 AND month < 10 THEN 3
  ELSE 4
  END quarter
FROM (
SELECT authorDimension.authorID, Article.articleTitle,
historyDimension.year, historyDimension.month
FROM authorDimension
JOIN ArticleAuthor
ON authorDimension.authorID = ArticleAuthor.authorID
JOIN Article
ON Article.articleID = ArticleAuthor.articleID
JOIN articleFact
ON Article.articleTitle = articleFact.articleTitle
JOIN historyDimension
ON articleFact.historyID = historyDimension.historyID
WHERE historyDimension.status = 'pubmed'
GROUP BY authorDimension.authorID,
historyDimension.year, historyDimension.month
)
GROUP BY authorID, year, quarter
"

# send the SQL query to the database
authorSum = dbGetQuery(dbcon, sqlCmd)
head(authorSum)
```

```
#library(RSQLite)
dbWriteTable(dbcon, "authorSummary", authorSum, append
= TRUE, row.names = FALSE)
```

Part 3.1 Queries showing seasonal pattern

Find number of articles published per quarter (note: all publication dates derived from history with ‘pubmed’ status)

```
library(sqldf)

sqlCmd = "SELECT Count(numArticles) as numArticles,
quarter
FROM journalSummary
GROUP BY quarter
"

# send the SQL query to the database
articlePerQuarter = dbGetQuery(dbcon, sqlCmd)
articlePerQuarter
```

Find number of articles by journal per quarter

```
library(sqldf)

sqlCmd = "SELECT Count(numArticles) as numArticles,
Journal.journalTitle, quarter
FROM journalSummary
JOIN Journal
ON Journal.journalID = journalSummary.journalID
GROUP BY Journal.journalTitle, quarter
```

```
"

# send the SQL query to the database
articlesByJournalPerQuarter = dbGetQuery(dbcon, sqlCmd)
articlesByJournalPerQuarter
```

Find number of Author contributions per quarter

```
library(sqldf)

sqlCmd = "SELECT Count(numArticles) as
numAuthorContributions, quarter
FROM authorSummary
JOIN Author
ON Author.authorID = authorSummary.authorID
GROUP BY quarter
"

# send the SQL query to the database
articlesByAuthorPerQuarter = dbGetQuery(dbcon, sqlCmd)
articlesByAuthorPerQuarter
```

Find average number of author contributions per quarter

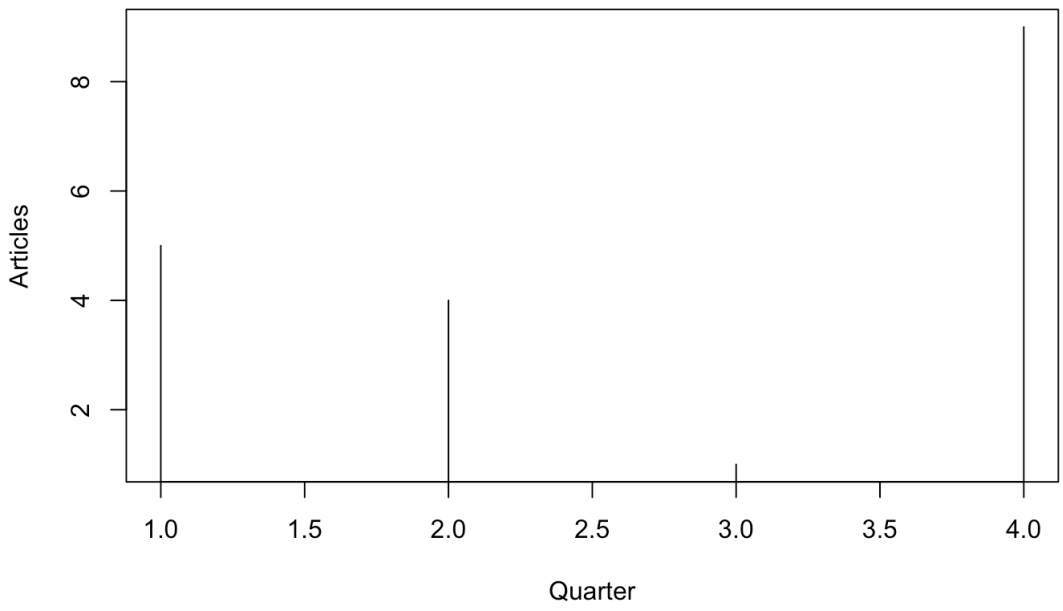
```
averageAuthPerQuart <- articlePerQuarter
averageAuthPerQuart$quarter <-
articlePerQuarter$quarter
averageAuthPerQuart$numAuthors <-
articlesByAuthorPerQuarter$numAuthorContributions
averageAuthPerQuart$averageAuthorContributions <-
averageAuthPerQuart$numAuthors /
averageAuthPerQuart$numArticles
```

```
averageAuthPerQuart$numArticles <- NULL
averageAuthPerQuart$numAuthors <- NULL

averageAuthPerQuart
```

Part 3.2 Visualizing data from queries

Number of articles per quarter

```
plot(x = articlePerQuarter$quarter, y =
articlePerQuarter$numArticles, type = "h",
xlab="Quarter", ylab="Articles",)
```

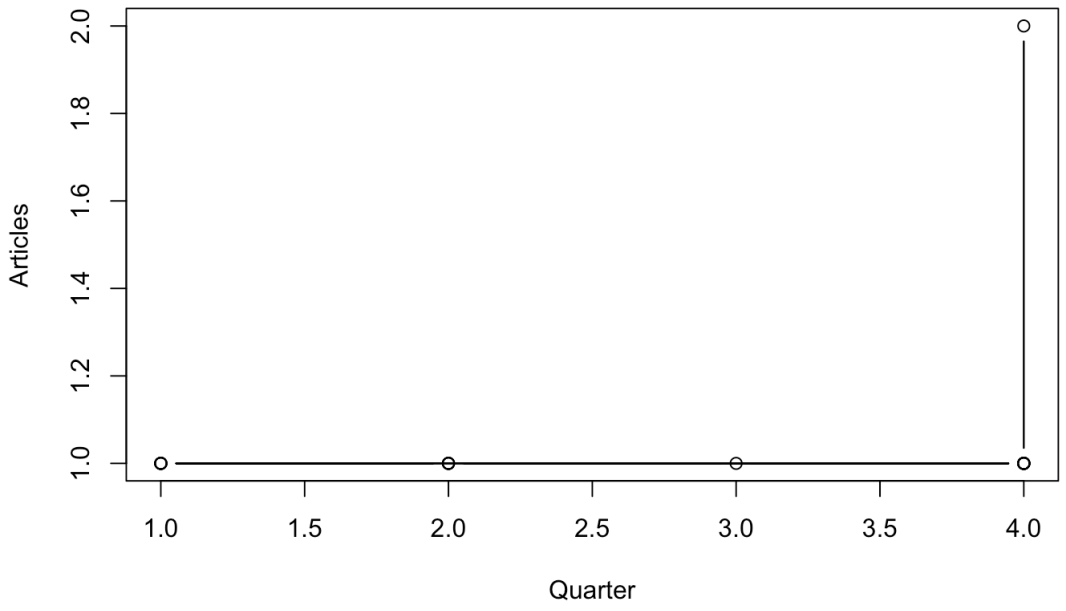


Quarter 4 has the most article publications (with a status of pubmed)

Number of articles by journal per

quarter

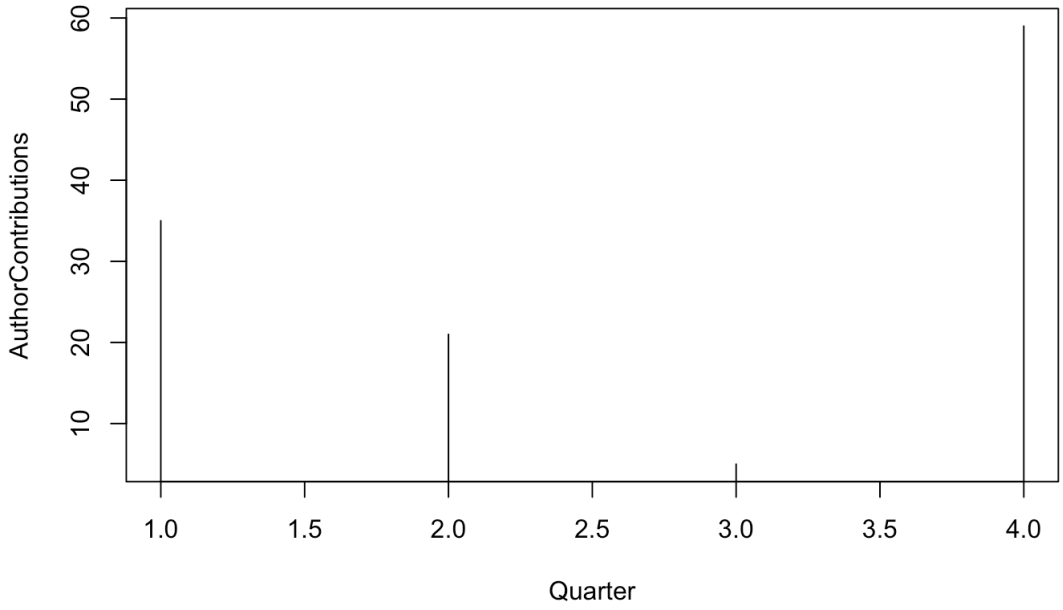
```
plot(x = articlesByJournalPerQuarter$quarter, y =
articlesByJournalPerQuarter$numArticles, type = "b",
xlab="Quarter", ylab="Articles",)
```



Only one journal has an article publication twice in the same quarter

Number of author contributions per quarter

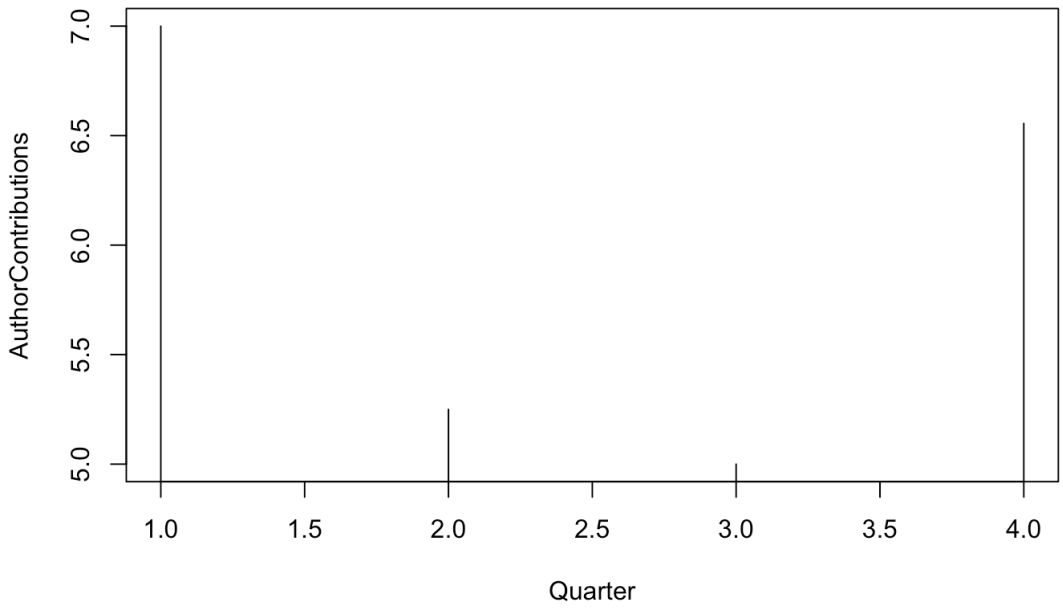
```
plot(x = articlesByAuthorPerQuarter$quarter, y =
articlesByAuthorPerQuarter$numAuthorContributions, type
= "h", xlab="Quarter", ylab="AuthorContributions",)
```



Most author contributions happen in the 4th quarter while the least happen in the third

Average number of author contributions per article per quarter

```
plot(x = averageAuthPerQuart$quarter, y =
averageAuthPerQuart$averageAuthorContributions, type =
"h", xlab="Quarter", ylab="AuthorContributions",)
```



The first quarter has the highest average of author contributions per article