Transcript Fee Request System

Megan Cusey

MSD 523 Spring B 2019

Table of Contents

[Section 1: Requirements 6](#_Toc9363602)

[Business Description 6](#_Toc9363603)

[Business Requirements 7](#_Toc9363604)

[Database Requirements 7](#_Toc9363605)

[Section 2: Data Models 9](#_Toc9363606)

[Logical Data Model 9](#_Toc9363607)

[Conceptual Data Model 9](#_Toc9363608)

[Section 3: Physical Database 14](#_Toc9363609)

[Tables 14](#_Toc9363610)

[Judge 15](#_Toc9363611)

[UserType 15](#_Toc9363612)

[Status 16](#_Toc9363613)

[Accounting 16](#_Toc9363614)

[Users 17](#_Toc9363615)

[Courthouse 18](#_Toc9363616)

[Transcript Fee Request 19](#_Toc9363617)

[Transcript Fee Request History 20](#_Toc9363618)

[Section 4: Queries 21](#_Toc9363619)

[Simple Queries 21](#_Toc9363620)

[Complex Queries 24](#_Toc9363621)

[Section 5: Data Warehouse Design and Coding 28](#_Toc9363622)

[Data Warehouse Design 28](#_Toc9363623)

[Data Marts 28](#_Toc9363624)

[Data Warehouse Creation Code 28](#_Toc9363625)

[Data Warehouse Queries Code 28](#_Toc9363626)

# Section 1: Requirements

## Business Description

At the Illinois Comptroller’s Office (IOC), we are tasked with processing payments for the state. As you can imagine, there are many different types of payments for various reasons that get issued by our office. In this study, IOC is taking on the payments made to court transcribers for indigent defendants. As a result, the web development team has been asked to build a web application that accepts a data entry form documenting these transcript fee requests. The business process is as follows:

* A court transcriber submits their request by completing the data entry form online
* The request is forwarded to the supervisor that overseas the courthouse the services were performed at
* The courthouse supervisor reviews and chooses to either approve or reject the transcript fee request
* If approved, the transcript fee request is assigned to the Court Reporting Services (CRS) Department at the IOC for final review
* If denied by the courthouse supervisor or CRS, the transcript fee request is returned for correction to the court transcriber
* If the transcript fee request is approved by CRS, the transcript fee request is forwarded to the IOC accounting staff for payment processing
* The IOC accounting staff runs a job in the web application development that extracts all the required information for payment which generates a file that is sent to the payment processing software. When the transcript fee request is included on the file, the request’s status is updated to Filed to Statewide Accounting Management System.
* The web application receives a file back from the Statewide Accounting Management System. When the transcript fee request is recorded on that file, the request’s status is updated to issued.
* The court transcriber can update their own transcript fee request as Rejected in the case that the request was mistakenly entered twice or inaccurately.

## Business Requirements

1. The Transcript Fee Request System must store all necessary payment information required to process payment to a court transcriber
2. The Transcript Fee Request System must store all necessary transcript request information to sufficiently identify valid transcript fee requests
3. The Transcript Fee Request System must track the status of the transcript fee request
4. The Transcript Fee Request System must assign the appropriate accounting codes per Fiscal Year
5. When the status of a Transcript Fee Request System changes, record to appropriate user the request is assigned to

## Database Requirements

1. The following information is required for a transcript fee request:
   1. Transcriber
   2. Date service was provided
   3. The amount of the request
   4. The judge who ordered it
   5. The courthouse the service was performed in
   6. The defendant name(s)
   7. Number of pages transcribed
2. The following information is required per transcriber:
   1. Payee Name
      1. Last Name, First Name, Middle Initial if individual
      2. Business Name if filing taxes with a TIN
   2. Tax Identification Number
      1. Social Security number if individual
      2. TIN if business
   3. Payee Address
   4. Payee City
   5. Payee State
   6. Payee Zip Code
3. The possible statuses for the transcript are:
   1. Submitted to Circuit Supervisor for Approval
   2. Submitted to Court Reporting Services for Approval
   3. Submitted to Illinois Comptroller’s Office for Processing
   4. Submitted to the Statewide Accounting Management System
   5. Payment Issued
   6. Rejected

# Section 2: Data Models

## Logical Data Model

## Conceptual Data Model

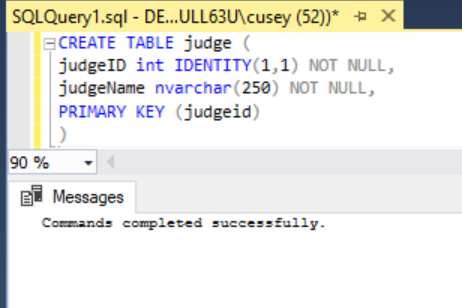
There are 6 major entities for the Transcript Fee Request Database. Details are below:

* **Accounting**: This table tracks what codes are sent to the payment processing system to assign the appropriate payment codes for the IOC’s books. Codes are assigned each Fiscal Year.
  + Fiscal Year
  + Object Code
  + Agency Code
  + Agency Name
* **Transcript Fee Request:** This tables records the current information for a transcript fee request, the status of the request, and who is currently assigned to the request.
  + Transcriber
  + Assigned To
  + Date of Service
  + Courthouse
  + Judge
  + Status
  + Defendant Name
  + Amount Requested
* **Transcript Fee Request History:** This tables records the historical information for a transcript fee request, the status of the request, and who updated the status of the request.
  + Request ID (FK)
  + Status Changed By
  + Previous Status
  + Status Changed Date
  + Date of Service
  + Courthouse
  + Judge
  + Status
  + Defendant Name
  + Amount Requested
* **Users:** This tables records the demographic information and user type for all users in the Transcript Fee Request system.
  + User Type ID (FK)
  + TIN
  + Payee Address 1
  + Payee Address 2
  + Payee City
  + Payee State
  + Payee Zip Code
  + Business Name
  + First Name
  + Middle Initial
  + Last Name
* **User Type:** This tables describes all possible user types.
  + User Type
* **Status:** This tables describes all possible statuses a transcript request may have.
  + Status
* **Courthouse:** This tables lists details about each courthouse
  + Courthouse Name
  + Address 1
  + Address 2
  + City
  + State
  + Zip
  + Supervisor
* **Judge:** This tables lists all judges.
  + Judge Name
  + **Transcript Fee Request History:** This tables records the historical information for a transcript fee request, the status of the request, and who updated the status of the request.
  + Request ID (FK)
  + Status Changed By
  + Previous Status
  + Status Changed Date
  + Date of Service
  + Courthouse
  + Judge
  + Status
  + Defendant Name

# Section 3: Physical Database

## Tables

### Judge



CREATE TABLE judge (

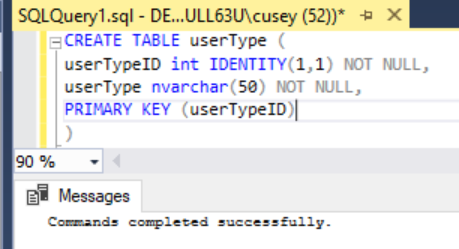
judgeID int IDENTITY(1,1) NOT NULL,

judgeName nvarchar(250) NOT NULL,

PRIMARY KEY (judgeid)

)

### UserType



CREATE TABLE userType (

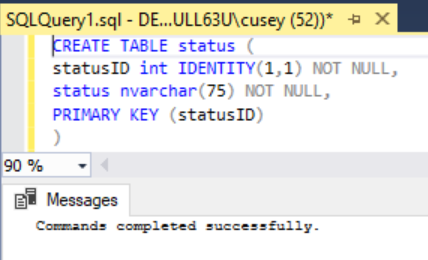
userTypeID int IDENTITY(1,1) NOT NULL,

userType nvarchar(50) NOT NULL,

PRIMARY KEY (userTypeID)

)

### Status



CREATE TABLE status (

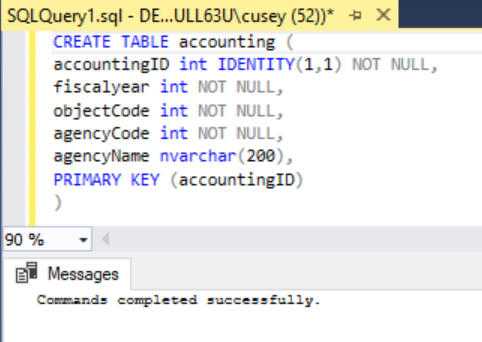
statusID int IDENTITY(1,1) NOT NULL,

status nvarchar(75) NOT NULL,

PRIMARY KEY (statusID)

)

### Accounting



CREATE TABLE accounting (

accountingID int IDENTITY(1,1) NOT NULL,

fiscalyear int NOT NULL,

objectCode int NOT NULL,

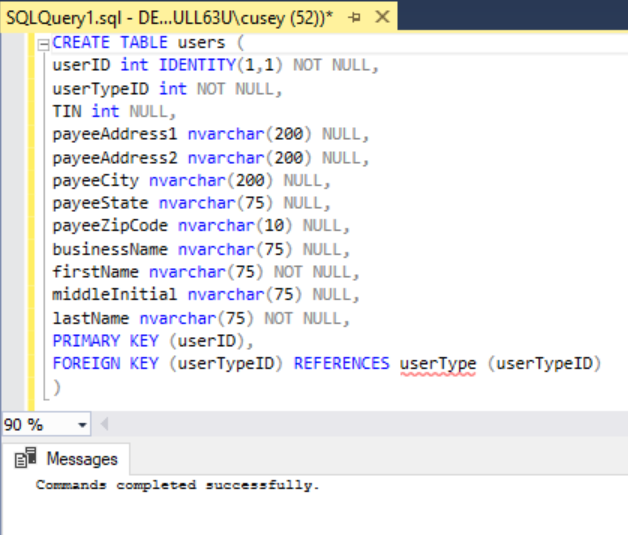
agencyCode int NOT NULL,

agencyName nvarchar(200),

PRIMARY KEY (accountingID)

)

### Users



CREATE TABLE users (

userID int IDENTITY(1,1) NOT NULL,

userTypeID int NOT NULL,

TIN int NULL,

payeeAddress1 nvarchar(200) NULL,

payeeAddress2 nvarchar(200) NULL,

payeeCity nvarchar(200) NULL,

payeeState nvarchar(75) NULL,

payeeZipCode nvarchar(10) NULL,

businessName nvarchar(75) NULL,

firstName nvarchar(75) NOT NULL,

middleInitial nvarchar(75) NULL,

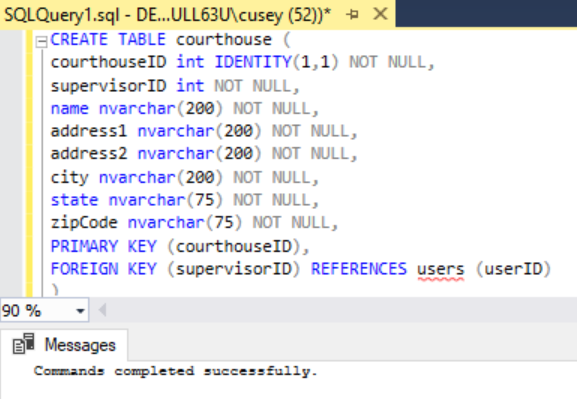
lastName nvarchar(75) NOT NULL,

PRIMARY KEY (userID),

FOREIGN KEY (userTypeID) REFERENCES userType (userTypeID)

)

### Courthouse



CREATE TABLE courthouse (

courthouseID int IDENTITY(1,1) NOT NULL,

supervisorID int NOT NULL,

name nvarchar(200) NOT NULL,

address1 nvarchar(200) NOT NULL,

address2 nvarchar(200) NOT NULL,

city nvarchar(200) NOT NULL,

state nvarchar(75) NOT NULL,

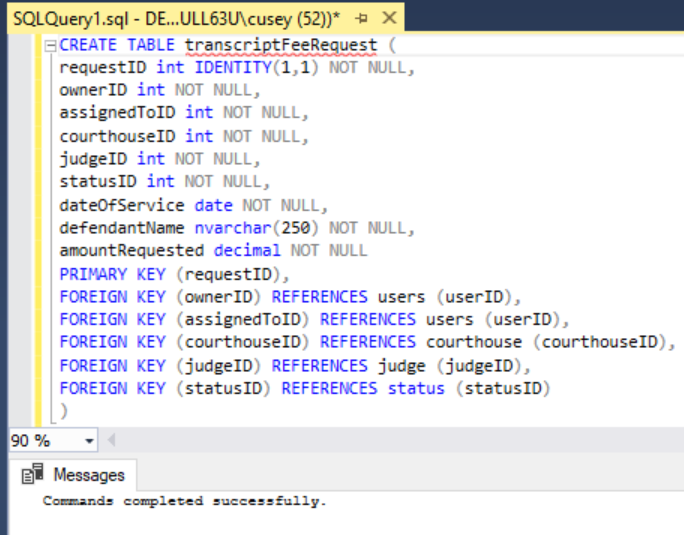
zipCode nvarchar(75) NOT NULL,

PRIMARY KEY (courthouseID),

FOREIGN KEY (supervisorID) REFERENCES users (userID)

)

### Transcript Fee Request



CREATE TABLE transcriptFeeRequest (

requestID int IDENTITY(1,1) NOT NULL,

ownerID int NOT NULL,

assignedToID int NOT NULL,

courthouseID int NOT NULL,

judgeID int NOT NULL,

statusID int NOT NULL,

dateOfService date NOT NULL,

defendantName nvarchar(250) NOT NULL,

amountRequested decimal NOT NULL

PRIMARY KEY (requestID),

FOREIGN KEY (ownerID) REFERENCES users (userID),

FOREIGN KEY (assignedToID) REFERENCES users (userID),

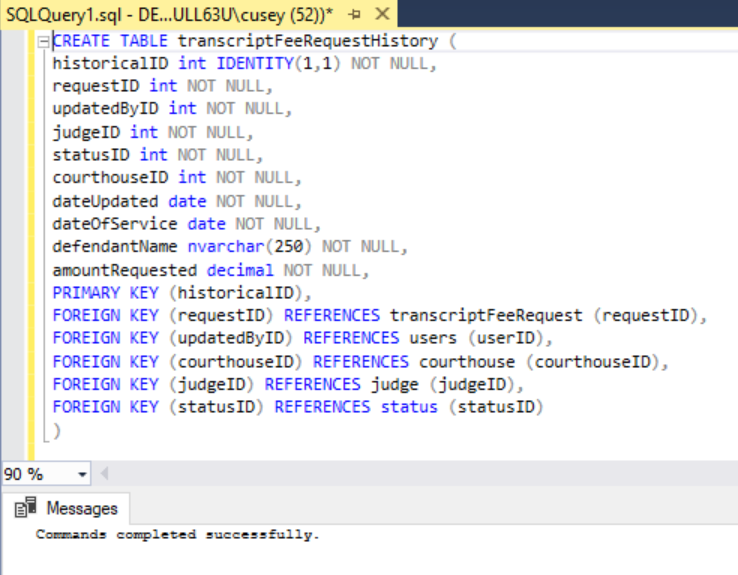
FOREIGN KEY (courthouseID) REFERENCES courthouse (courthouseID),

FOREIGN KEY (judgeID) REFERENCES judge (judgeID),

FOREIGN KEY (statusID) REFERENCES status (statusID)

)

### Transcript Fee Request History



CREATE TABLE transcriptFeeRequestHistory (

historicalID int IDENTITY(1,1) NOT NULL,

requestID int NOT NULL,

updatedByID int NOT NULL,

judgeID int NOT NULL,

statusID int NOT NULL,

courthouseID int NOT NULL,

dateUpdated date NOT NULL,

dateOfService date NOT NULL,

defendantName nvarchar(250) NOT NULL,

amountRequested decimal NOT NULL,

PRIMARY KEY (historicalID),

FOREIGN KEY (requestID) REFERENCES transcriptFeeRequest (requestID),

FOREIGN KEY (updatedByID) REFERENCES users (userID),

FOREIGN KEY (courthouseID) REFERENCES courthouse (courthouseID),

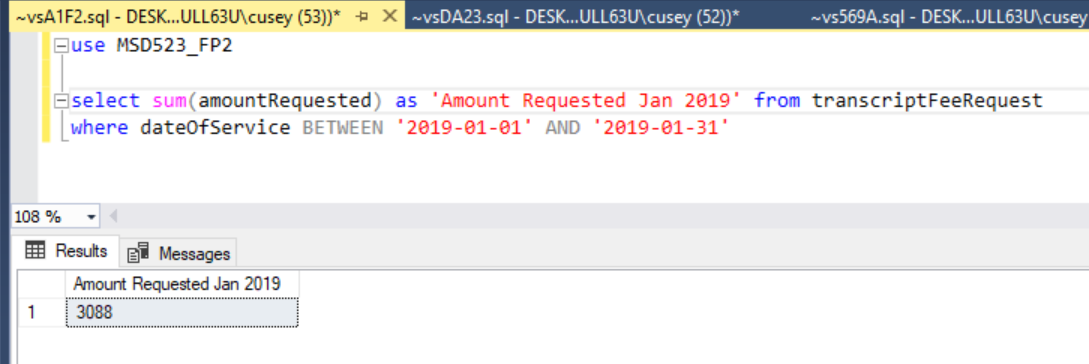
FOREIGN KEY (judgeID) REFERENCES judge (judgeID),

FOREIGN KEY (statusID) REFERENCES status (statusID)

)

# Section 4: Queries

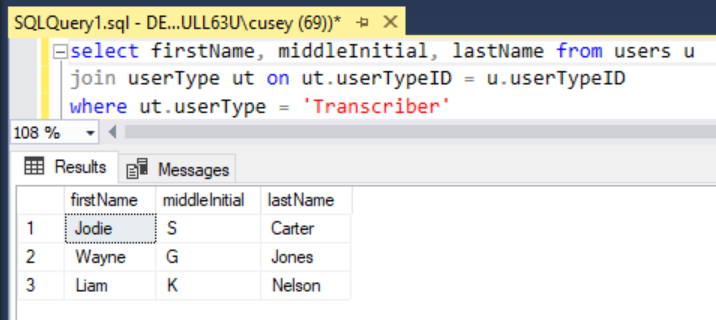
## Simple Queries

1. This query provides the total amount requested by transcripts in between two dates. In this example, I queried for the sum of the requested in January of 2019

select sum(amountRequested) as 'Amount Requested Jan 2019' from transcriptFeeRequest

where dateOfService BETWEEN '2019-01-01' AND '2019-01-31'

1. The below query returns a list of Transcriber’s names.

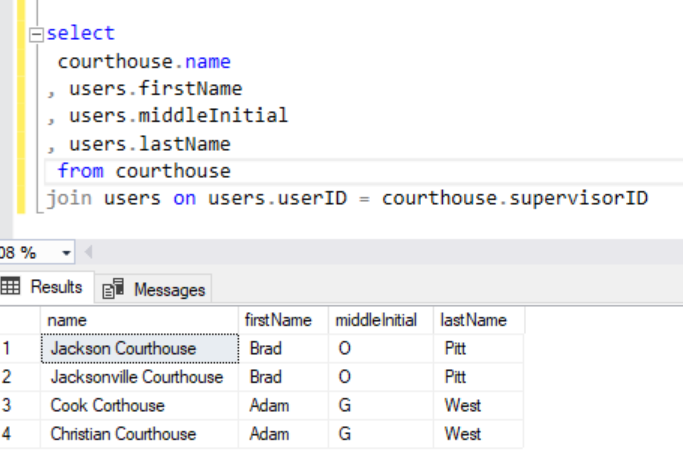


select firstName, middleInitial, lastName from users u

join userType ut on ut.userTypeID = u.userTypeID

where ut.userType = 'Transcriber'

1. The following query identifies the supervisor that belongs to a courthouse



select

courthouse.name

, users.firstName

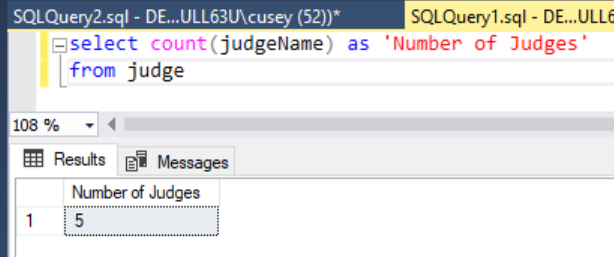
, users.middleInitial

, users.lastName

from courthouse

join users on users.userID = courthouse.supervisorID

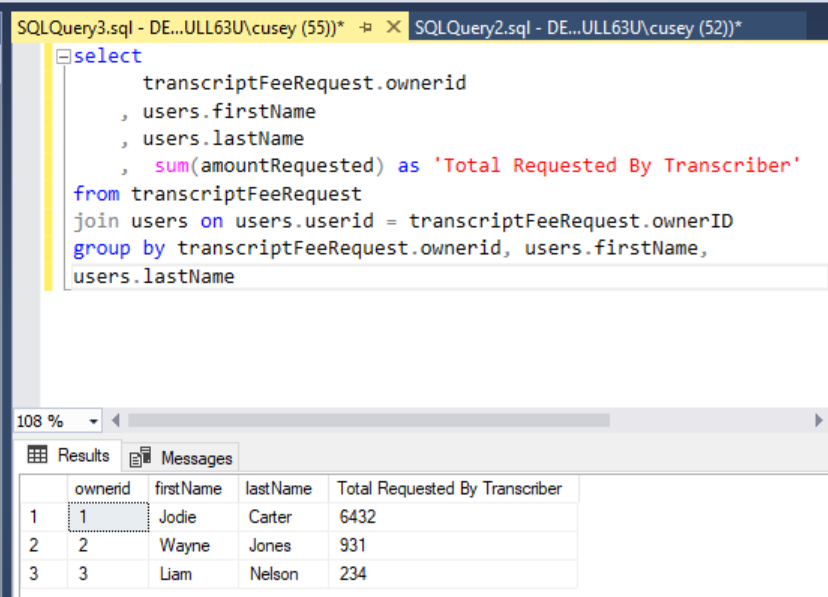
1. The following returns a count of judges



select count(judgeName) as 'Number of Judges'

from judge

1. The following query returns the total amount requested per transcriber



## Complex Queries

1. The following query provides all relevant current details of all transcript fee requests. This includes the current status, who the request is assigned to, who owns the request, the courthouse, date, and judge the services where provided, and amount of payment due.



select

t.requestID

, CONCAT(u2.lastname,' ',u2.firstName,' ', u2.middleInitial) as 'Owner'

, CONCAT(u.lastname,' ',u.firstName,' ', u.middleInitial) as 'Assigned To'

, c.name as 'Courthouse'

, j.judgeName as 'Judge'

, s.status

, t.dateOfService

, t.defendantName

, t.amountRequested

from transcriptFeeRequest t

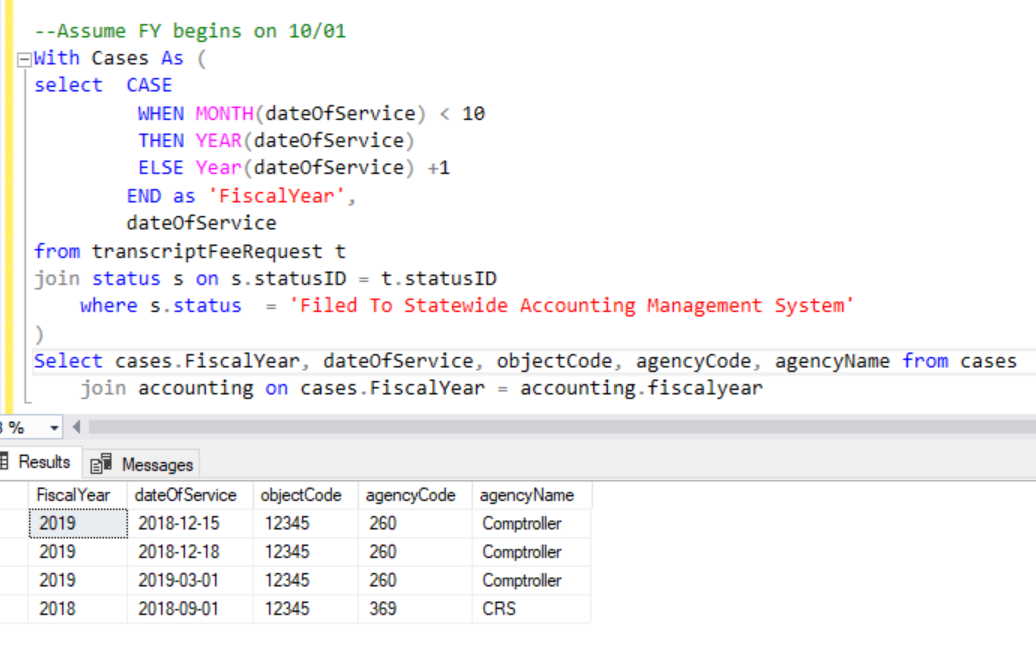
join status s on s.statusID = t.statusID

join users u on u.userID = t.assignedToID

join users u2 on u2.userID = t.ownerid

join courthouse c on c.courthouseID = t.courthouseID

join judge j on j.judgeID = t.judgeID

1. This query retrieves the fiscal year for each date of service and the associated accounting codes for the payments that have been filed to the Statewide Accounting Management System.

--Assume FY begins on 10/01

With Cases As (

select CASE

WHEN MONTH(dateOfService) < 10

THEN YEAR(dateOfService)

ELSE Year(dateOfService) +1

END as 'FiscalYear',

dateOfService

from transcriptFeeRequest t

join status s on s.statusID = t.statusID

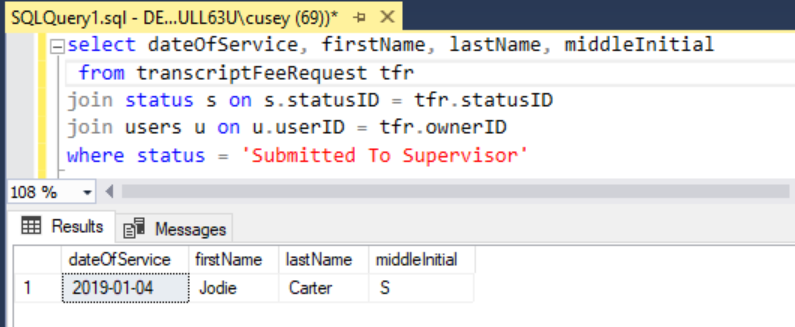
where s.status = 'Filed To Statewide Accounting Management System'

)

Select cases.FiscalYear, dateOfService, objectCode, agencyCode, agencyName from cases

join accounting on cases.FiscalYear = accounting.fiscalyear

1. The query returns transcript fee request’s data of service and who requested the payment.



select dateOfService, firstName, lastName, middleInitial

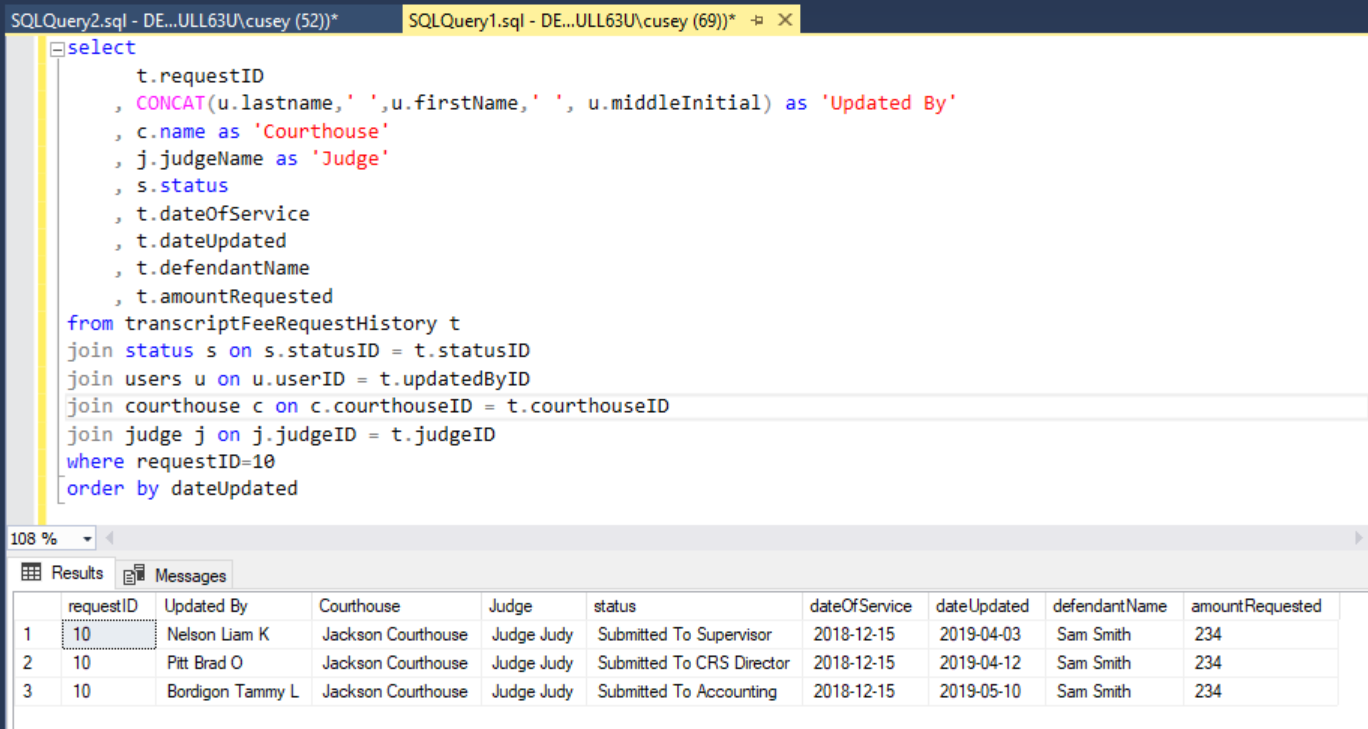
from transcriptFeeRequest tfr

join status s on s.statusID = tfr.statusID

join users u on u.userID = tfr.ownerID

where status = 'Submitted To Supervisor'

1. The following query provides the historical status information of a specified request.



select

t.requestID

, CONCAT(u.lastname,' ',u.firstName,' ', u.middleInitial) as 'Updated By'

, c.name as 'Courthouse'

, j.judgeName as 'Judge'

, s.status

, t.dateOfService

, t.dateUpdated

, t.defendantName

, t.amountRequested

from transcriptFeeRequestHistory t

join status s on s.statusID = t.statusID

join users u on u.userID = t.updatedByID

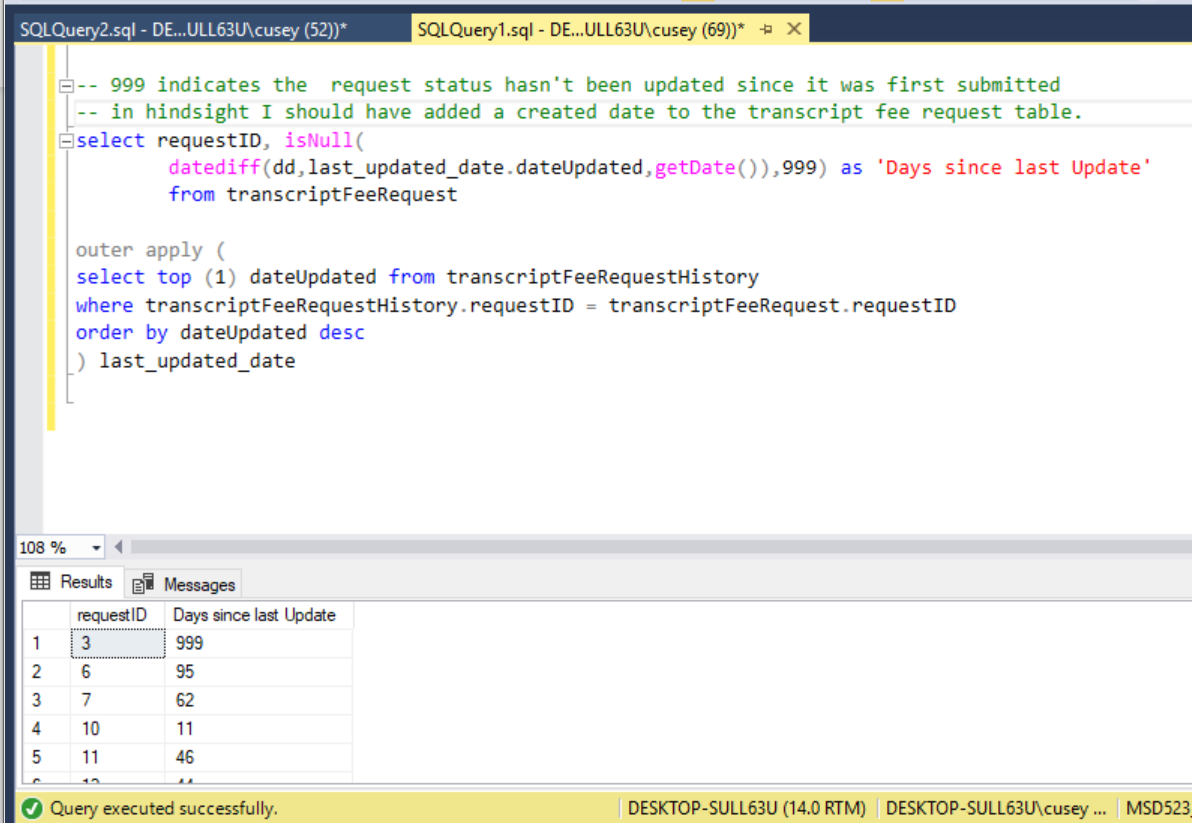
join courthouse c on c.courthouseID = t.courthouseID

join judge j on j.judgeID = t.judgeID

where requestID=10

order by dateUpdated

1. The following provides the number of days since the transcript fee request was last updated.



-- 999 indicates the request status hasn't been updated since it was first submitted

-- in hindsight I should have added a created date to the transcript fee request table.

select requestID, isNull(

datediff(dd,last\_updated\_date.dateUpdated,getDate()),999) as 'Days since last Update'

from transcriptFeeRequest

outer apply (

select top (1) dateUpdated from transcriptFeeRequestHistory

where transcriptFeeRequestHistory.requestID = transcriptFeeRequest.requestID

order by dateUpdated desc

) last\_updated\_date

# Section 5: Data Warehouse Design and Coding

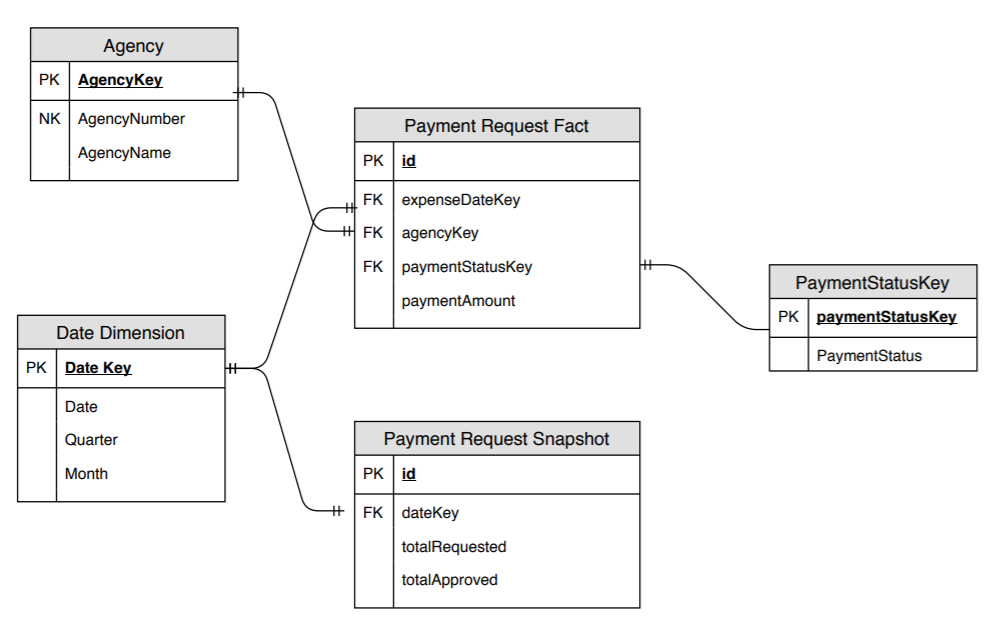
## Data Warehouse Design

The Comptroller’s Office has many similar applications to the Transcript Fee Request system that processes payments on behalf of the state’s agencies. Perhaps one of the biggest issues is determining the amount of cash we can pay out versus the number of requests. In the data warehouse design is based on extracting, transforming, and loading the payment details from every web application and getting that information into one place in order to perform analysis.

A high-level overview of the business process is that agencies and/or individuals performing services for the state request payment. For example, one of the largest payment processing systems we have is for payroll. Upon verifying the validity of the payments, the office runs the information through the Statewide Accounting Management System which allows the payments to be issued. The purpose for this data warehouse is to extract the payment details while still in a pending state to analyze the future cash at hand amount before it reaches the accounting staff.

The grain of the data warehouse fact table is one row per payment request. The dimensions will be Agency, Date, and Payment Status. The Agency Dimension will include Agency Key, Agency Number, and Agency Name. The Date Dimension will include Date, Quarter, and Month. The Payment Status Dimension will include Payment Status Key and Payment Status. The Payment Status Request will indicate that the payment hasn’t been vetted yet. The payment status of Approved will indicate that it’s waiting to be inserted into the accounting system.

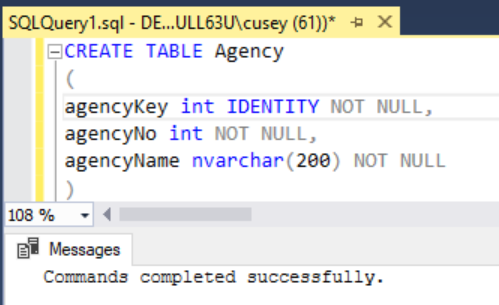
There are two fact tables, Payment Request Facts and Payment Request Snapshot. The Payment Request Facts includes expenseDateKey, the day the service was provided/expensure incurred, agencyKey (the cost center), payment status key, and the payment amount. The Payment Request Snapshot will include one row per day (date key), with the sum of the total amount requested, and the sum of the total amount approved. This will allow for a quick, high level overview of the potential and promised liabilities to come.

\\

## Data Marts

I do not think a data mart is necessary in this instance. The data warehouse schema isn’t too big. Eventually, the table rows will get extensive for each payment request, however, the data can be purged every couple of years when the information isn’t very relevant.

## Data Warehouse Creation Code



CREATE TABLE Agency

(

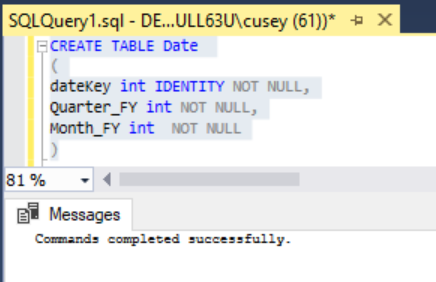
agencyKey int IDENTITY NOT NULL,

agencyNo int NOT NULL,

agencyName nvarchar(200) NOT NULL

)

ALTER TABLE [dbo].agency WITH CHECK ADD PRIMARY KEY(agencyKey)



CREATE TABLE Date

(

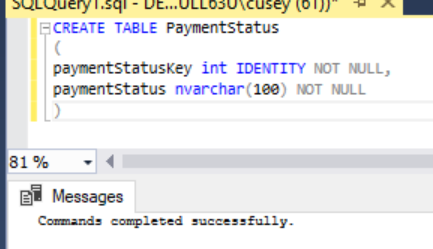
dateKey int IDENTITY NOT NULL,

Quarter\_FY int NOT NULL,

Month\_FY int NOT NULL

Date\_FY date NOT NULL) – Added via designer after I realized I missed this column

ALTER TABLE [dbo].date WITH CHECK ADD PRIMARY KEY(dateKey)



CREATE TABLE PaymentStatus

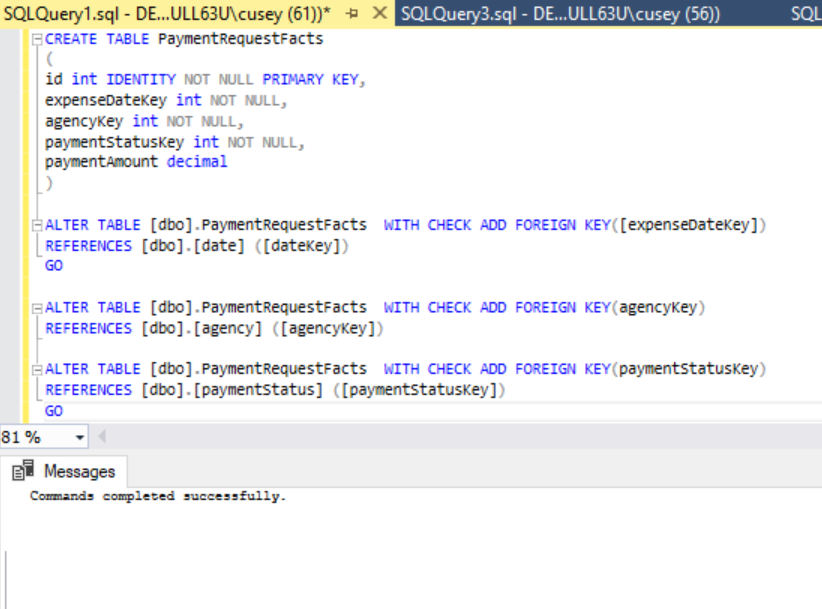
(

paymentStatusKey int IDENTITY NOT NULL,

paymentStatus nvarchar(100) NOT NULL

)

ALTER TABLE [dbo].paymentStatus WITH CHECK ADD PRIMARY KEY(paymentStatusKey)



CREATE TABLE PaymentRequestFacts

(

id int IDENTITY NOT NULL PRIMARY KEY,

expenseDateKey int NOT NULL,

agencyKey int NOT NULL,

paymentStatusKey int NOT NULL,

paymentAmount decimal

)

ALTER TABLE [dbo].PaymentRequestFacts WITH CHECK ADD FOREIGN KEY([expenseDateKey])

REFERENCES [dbo].[date] ([dateKey])

GO

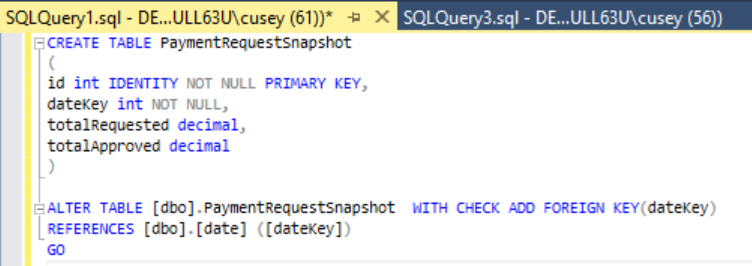
ALTER TABLE [dbo].PaymentRequestFacts WITH CHECK ADD FOREIGN KEY(agencyKey)

REFERENCES [dbo].[agency] ([agencyKey])

ALTER TABLE [dbo].PaymentRequestFacts WITH CHECK ADD FOREIGN KEY(paymentStatusKey)

REFERENCES [dbo].[paymentStatus] ([paymentStatusKey])

GO



CREATE TABLE PaymentRequestSnapshot

(

id int IDENTITY NOT NULL PRIMARY KEY,

dateKey int NOT NULL,

totalRequested decimal,

totalApproved decimal

)

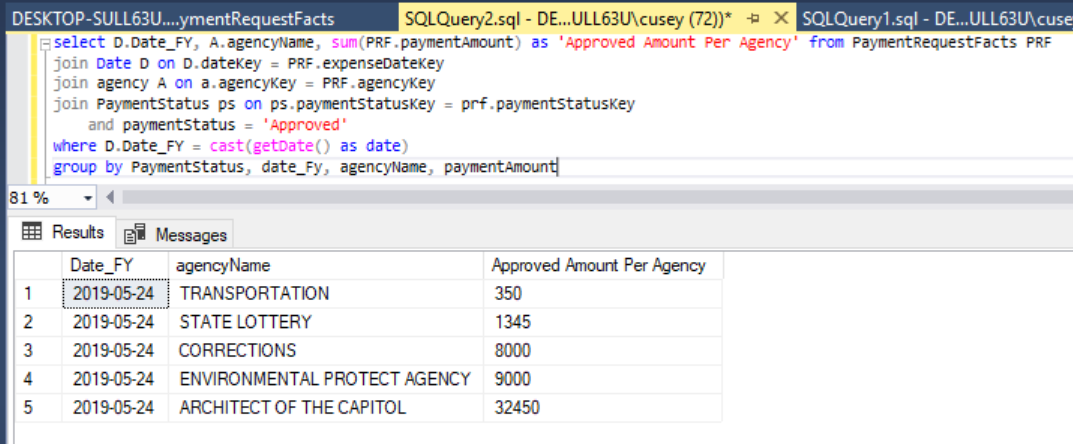
ALTER TABLE [dbo].PaymentRequestSnapshot WITH CHECK ADD FOREIGN KEY(dateKey)

REFERENCES [dbo].[date] ([dateKey])

GO

## Data Warehouse Queries Code

1. Retrieve the Approved Amount for all Agencies for the current date:



select D.Date\_FY, A.agencyName, sum(PRF.paymentAmount) as 'Approved Amount Per Agency' from PaymentRequestFacts PRF

join Date D on D.dateKey = PRF.expenseDateKey

join agency A on a.agencyKey = PRF.agencyKey

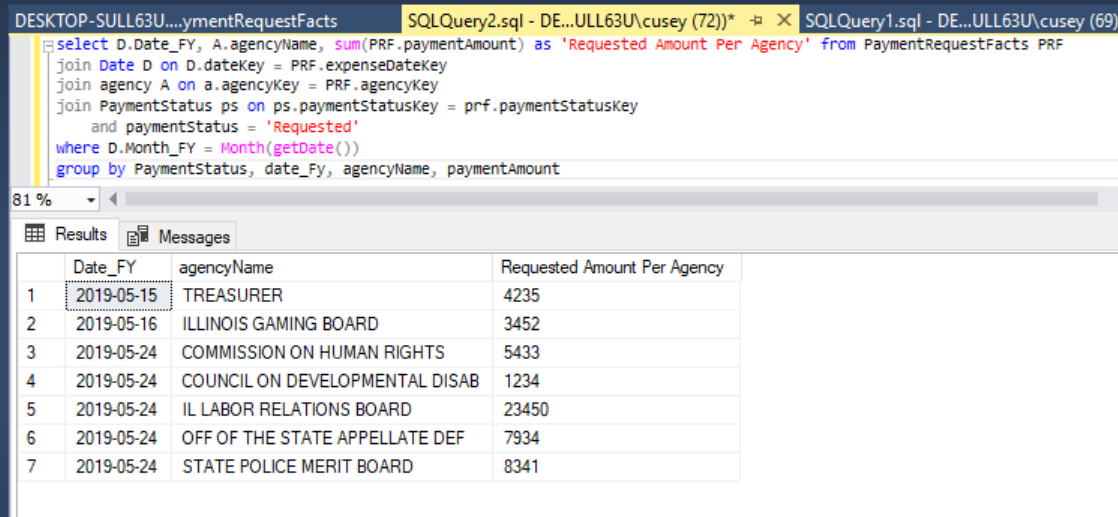
join PaymentStatus ps on ps.paymentStatusKey = prf.paymentStatusKey

and paymentStatus = 'Approved'

where D.Date\_FY = cast(getDate() as date)

group by PaymentStatus, date\_Fy, agencyName, paymentAmount

1. Get the Requested Amount for the entire month per Agency.



select D.Date\_FY, A.agencyName, sum(PRF.paymentAmount) as 'Requested Amount Per Agency' from PaymentRequestFacts PRF

join Date D on D.dateKey = PRF.expenseDateKey

join agency A on a.agencyKey = PRF.agencyKey

join PaymentStatus ps on ps.paymentStatusKey = prf.paymentStatusKey

and paymentStatus = 'Requested'

where D.Month\_FY = Month(getDate())

group by PaymentStatus, date\_Fy, agencyName, paymentAmount

1. Retrieve the amount approved and amount requested for today

