Analyzing Health Data from a College Dorm

Joe Ellis, Christina Engstrom and Eric Weber

May 4, 2016

Contents

1 Introduction

1.1 Background and Description of Dataset

As the ability to gather and store data increases exponentially, we are able to ask and gain real time insight into questions about which we could not conceive just a few years ago. A prime example of this space is understanding how people interact, move, and change over small periods of time. Our primary aim in this project is to use an existing dataset about students' movements and interactions in a single dorm on a single college campus to create a data warehouse that would provide insight into students' previous behavior and serve as a real time predictor of issues or changes relevant to students' lives.

The dataset for this project comes from the Human Dynamics Lab at MIT. The dataset itself requires a request to the lab, but is open to anyone interested. The request for the dataset serves as a check and commitment that one will not try to identify the students using the data. The data describes the spatial-temporal patterns of residents in a single MIT dorm in the academic year 2008 to 2009. Their movement and interactions were tracked every six minutes during the course of the study using mobile phone data, and the students self reported about other measures: music interest, political thinking, height, weight, illnesses, and more. There are a number of CSV files containing different measures at different points in time and the files are linked by a subject id number. That link provides us the ability to tie relevant records together in a data warehouse.

1.2 Business Requirements and Analysis Questions

The business lens that we are assuming in constructing this data warehouse is that of a residence hall director, who would be interested in knowing general de-identified information or trends about the students living in university housing. Currently hall directors rely primarily on word of mouth or students proactively reaching out to gather pertinent information on their student population. The warehouse that we build will contain information that a hall director may be interested in, such as self-reported health and disease data, dorm and floor information, and students' interests, in an accessible format. We want the hall directors to be able to use this data warehouse to gather actionable information so that they can be proactive in identifying and responding to trends. This information would allow them to answer questions such as:

- Are students getting progressively healthier or unhealthier throughout the year?
- Do healthy students stay healthy? (and vice versa)
- How and where do flu symptoms spread?
- What trends can we see in students' interests and activities?

While these questions proved to be beyond the scope of this particular project, the mechanisms we created could serve as a foundation for expansion of the database. In this report, we focus on the construction and evaluation of a table focused on the health measures of students over the course of the academic year. We focus on these issues in two parts: first providing a detail description of the construction of the fact table, and second, describing our analyses of the data from the fact table to identify important patterns in the dataset.

2 Building the Fact Table

This section focuses on the creation of the fact table. The purpose of the fact table was to enable analysis of health related questions as described in the previous section, but without the typical normalized database structure. We determined the fact table would have four dimensions: survey month, smoking status, diet status and user identification. Each instance of the fact contained information on the user such as weight, height, salads and fruits consumed and activity levels each week. Below is an example of the raw data file which we used. Even these ten lines suggest issues with the data, such as strange values for weight, height and smoking status. We focus on those within our discussion of data merging in SSIS.

Table 1: First Lines of Raw Data

userId	wt	hght	salads	veggies	diet	aerobic	sports	smoking	time
55	140	69	0	0	Below average	2	0	Never	2008.09
36	150	67	2	1	Below average	3	3	Never	2008.09
39	105	66	0	2	Average	1	0	Never	2008.09
37	220	77	2	5	Very healthy	5	5	Never	2008.09
72	135	62	0	1	Unhealthy	0	0	Never	2008.09
11	180	69	0	1	Unhealthy	5	0	Never	2008.09
63	0	2801	7	0	Very unhealthy	0	1	0.000000	2008.09
21	240	75	0	1	Unhealthy	1	0	Never	2008.09
59	170	72	3	3	Average	5	5	Never	2008.09
47	145	68	1	3	Healthy	3	0	Never	2008.09

2.1 Creation of Script for Table Builds

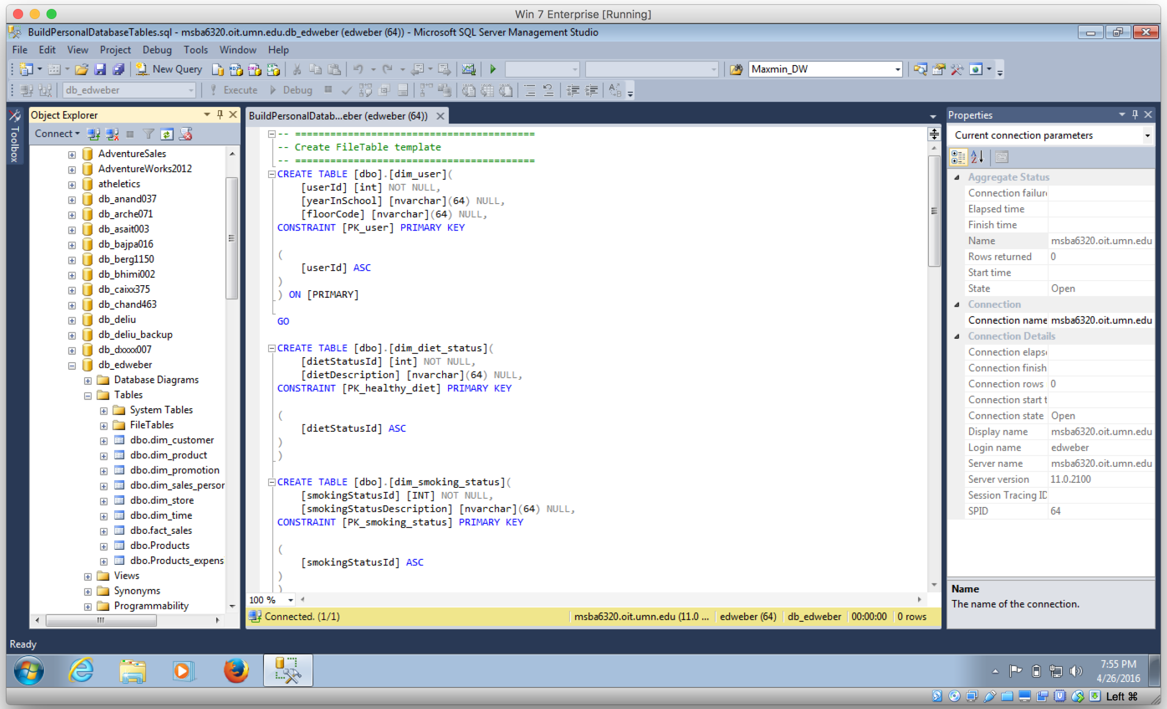
The first step in the ETL process was determining the form of the dimensions and the fact table. The SQL script below created five tables: the user, smoking status, diet status, and survey month dimensions, along with the fact table which had those four dimensions. This structure mirrored the dimension tables themselves, which we filled during the SSIS process.

```
CREATE TABLE [dbo].[dim_user](
    [userId] [int] NOT NULL,
    [yearInSchool] [nvarchar](64) NULL,
    [floorCode] [nvarchar](64) NULL,
CONSTRAINT [PK_user] PRIMARY KEY

(
    [userId] ASC
)
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[dim_diet_status](
    [dietStatusId] [int] NOT NULL,
    [dietDescription] [nvarchar](64) NULL,
CONSTRAINT [PK_healthy_diet] PRIMARY KEY

(
    [dietStatusId] ASC
)
)

CREATE TABLE [dbo].[dim_smoking_status](
    [smokingStatusId] [INT] NOT NULL,
    [smokingStatusDescription] [nvarchar](64) NULL,
CONSTRAINT [PK_smoking_status] PRIMARY KEY

(
    [smokingStatusId] ASC
)
)

CREATE TABLE [dbo].[dim_survey_month](
    [surveyMonthId] [nvarchar](64) NOT NULL,
    [surveyMonthDescription] [nvarchar](64) NULL,
CONSTRAINT [PK_survey_month] PRIMARY KEY

(
    [surveyMonthId] ASC
)
)

CREATE TABLE [dbo].[fact_health](
    [userId] [int] NULL,
    [yearInSchool] [nvarchar](64) NULL,
    [floorCode] [nvarchar](64) NULL,
    [surveyMonthId] [nvarchar](64) NULL,
    [surveyMonthDescription] [nvarchar](64) NULL,
    [smokingStatusId] [int] NULL,
    [smokingStatusDescription] [nvarchar](64) NULL,
    [dietStatusId] [int] NULL,
    [dietDescription] [nvarchar](64) NULL,
    [currentWeight] [int] NULL,
    [currentHeight] [int] NULL,
    [saladsPerWeek] [int] NULL,
    [veggiesFruitsPerWeek] [int] NULL,
    [aerobicPerWeek] [int] NULL,
    [sportsPerWeek] [int] NULL
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[fact_health]
WITH NOCHECK ADD  CONSTRAINT [Health_Information-User] FOREIGN KEY([userId])
REFERENCES [dbo].[dim_user] ([userId])
GO

ALTER TABLE [dbo].[fact_health] NOCHECK CONSTRAINT [Health_Information-User]
GO

ALTER TABLE [dbo].[fact_health]
WITH NOCHECK ADD  CONSTRAINT [Health_Information-Time] FOREIGN KEY([surveyMonthId])
REFERENCES [dbo].[dim_survey_month] ([surveyMonthId])
GO

ALTER TABLE [dbo].[fact_health] NOCHECK CONSTRAINT [Health_Information-Time]
GO

ALTER TABLE [dbo].[fact_health]
WITH NOCHECK ADD  CONSTRAINT [Health_Information-Smoking] FOREIGN KEY([smokingStatusId])
REFERENCES [dbo].[dim_smoking_status] ([smokingStatusId])
GO
```

```sql
ALTER TABLE [dbo].[fact_health] NOCHECK CONSTRAINT [Health_Information-Smoking]
GO

ALTER TABLE [dbo].[fact_health]
WITH NOCHECK ADD  CONSTRAINT [Health_Information-Diet] FOREIGN KEY([dietStatusId])
REFERENCES [dbo].[dim_diet_status] ([dietStatusId])
GO

ALTER TABLE [dbo].[fact_health] NOCHECK CONSTRAINT [Health_Information-Diet]
GO
```

2.2 Build Tables in SQL Server

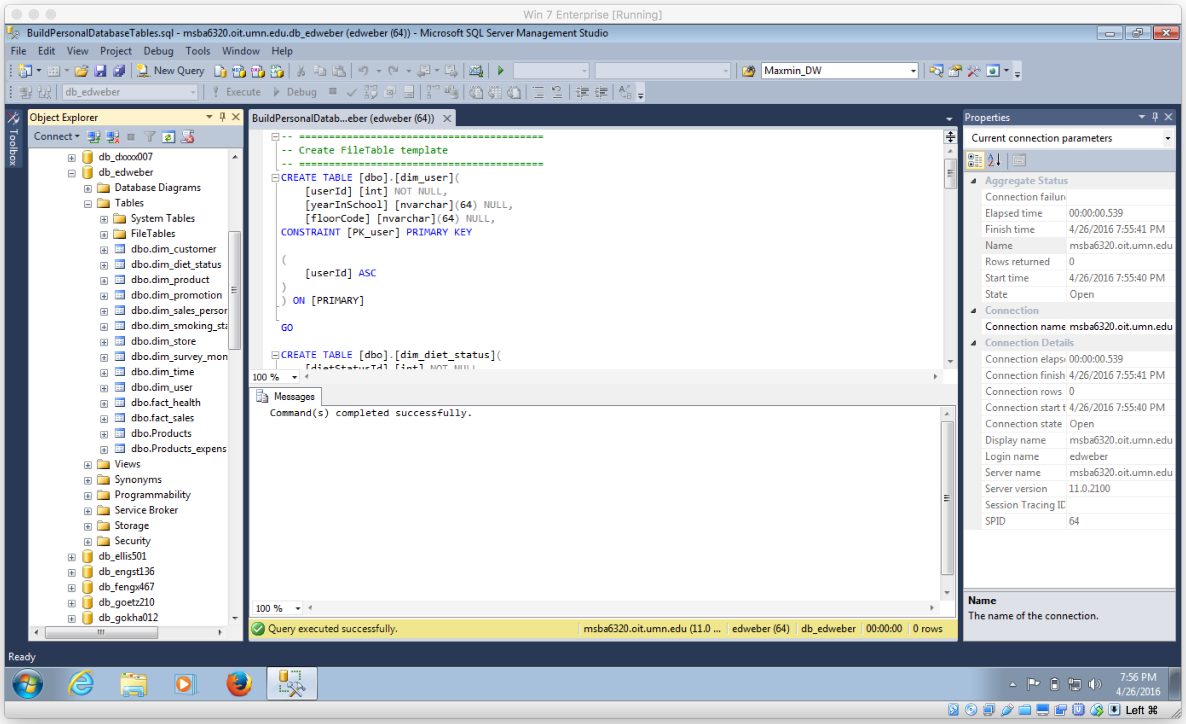
The next step of the ETL process was building the dimension and fact tables in the SQL Server, prior to any data being imported into the tables. The two figures below illustrate the before and after of executing the SQL script provided in the previous section. On the lefthand side it is clear that the script executed successfully and produced the desired tables in the user database. For reference, the connection to this database is denoted by WeberDB in future screen captures.



BuildPersonalDatabaseTables.sql - msba6320.oit.umn.edu.db_edweber (edweber (64)) - Microsoft SQL Server Management Studio

```
-- ========================================
-- Create FileTable template
-- ========================================
CREATE TABLE [dbo].[dim_user](
    [userId] [int] NOT NULL,
    [yearInSchool] [nvarchar](64) NULL,
    [floorCode] [nvarchar](64) NULL,
CONSTRAINT [PK_user] PRIMARY KEY

(
    [userId] ASC
)
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[dim_diet_status](
    [dietStatusId] [int] NOT NULL,
    [dietDescription] [nvarchar](64) NULL,
CONSTRAINT [PK_healthy_diet] PRIMARY KEY

(
    [dietStatusId] ASC
)
)

CREATE TABLE [dbo].[dim_smoking_status](
    [smokingStatusId] [INT] NOT NULL,
    [smokingStatusDescription] [nvarchar](64) NULL,
CONSTRAINT [PK_smoking_status] PRIMARY KEY

(
    [smokingStatusId] ASC
)
)
```

Connected. (1/1) | msba6320.oit.umn.edu (11.0 ... | edweber (64) | db_edweber | 00:00:00 | 0 rows



BuildPersonalDatabaseTables.sql - msba6320.oit.umn.edu.db_edweber (edweber (64)) - Microsoft SQL Server Management Studio

```
-- ========================================
-- Create FileTable template
-- ========================================
CREATE TABLE [dbo].[dim_user](
    [userId] [int] NOT NULL,
    [yearInSchool] [nvarchar](64) NULL,
    [floorCode] [nvarchar](64) NULL,
CONSTRAINT [PK_user] PRIMARY KEY

(
    [userId] ASC
)
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[dim_diet_status](
```

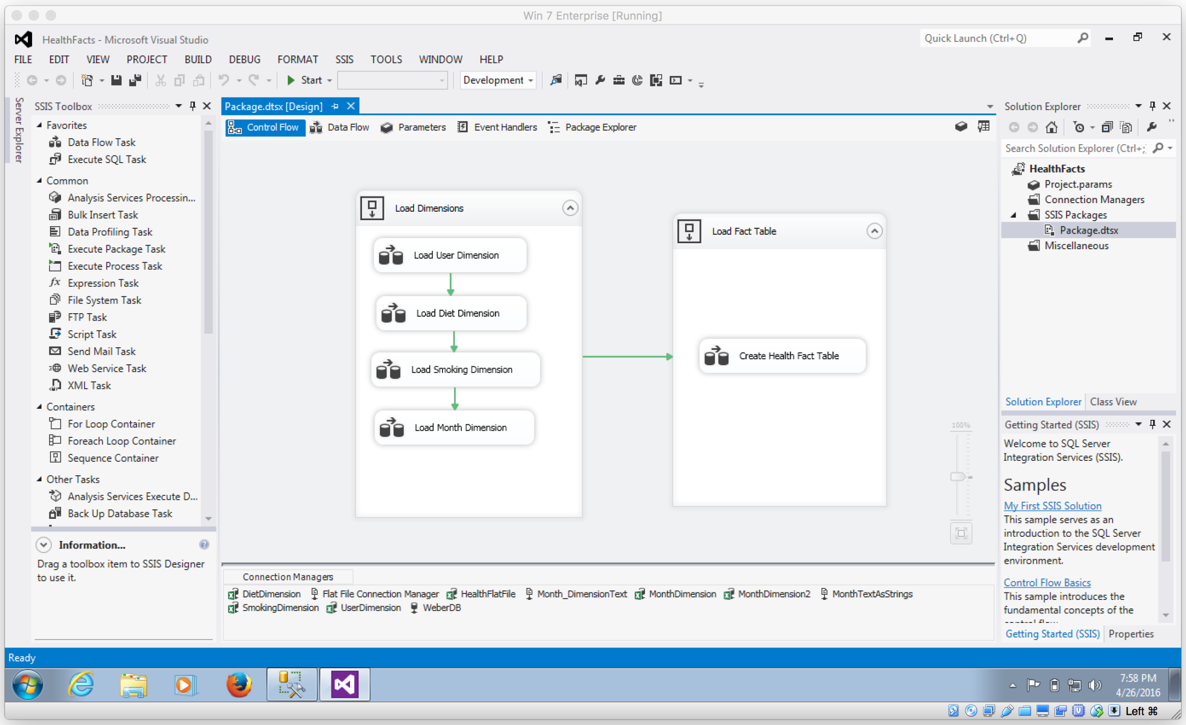
Messages

```
Command(s) completed successfully.
```

Query executed successfully. | msba6320.oit.umn.edu (11.0 ... | edweber (64) | db_edweber | 00:00:00 | 0 rows

2.3 SSIS Containers and Data Flows

Once we had the tables in place in SQL server, we constructed the data flows to fill those dimension tables and to create the fact table that we used for business analysis purposes. The figure below provides a basic overview of the logic behind the data flows. In the first container, we sequentially filled the dimension tables on the SQL server with data (see next few sections for details). Once that process completed, we filled the fact table with the raw data file and the created dimensions.



2.4 Dimensions

This section provides a basic overview of how we created the dimensions and displays the data we input into those dimension tables.

2.4.1 User Dimension

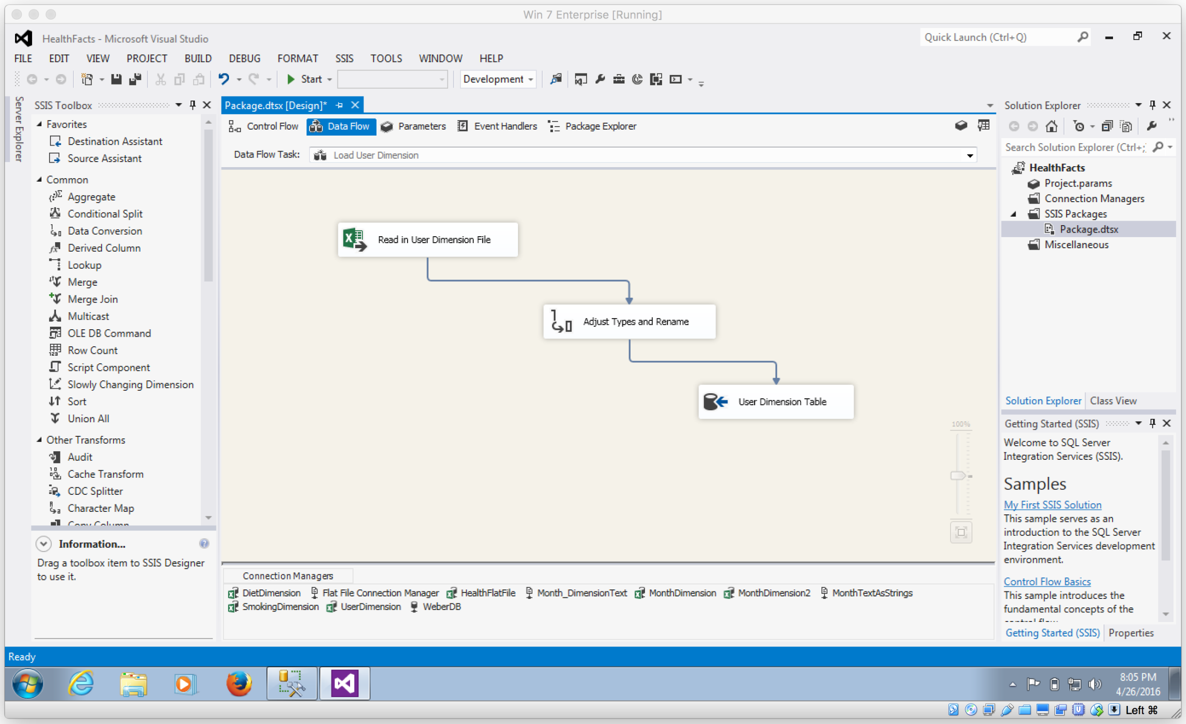
The user dimension came from another raw data file provided by the MIT Human Dynamics Lab. This file contains information on the userId, the floor on which the user lived (obviously a pseudo-identification) and the year in school of that user. The first ten lines of the user dimension are provided below. What is not obvious from these ten lines is that for some users, some or all of the floor and year in school information was missing.

```r
library(RODBC)
user_file <- sqlQuery(db_edweber, "SELECT *
                                   FROM   dim_user")
knitr::kable(user_file[c(1:10),],
             digits = 2,
             caption = "First Lines of User Dimension Table")
```

Table 2: First Lines of User Dimension Table

userId	yearInSchool	floorCode
1	GRT / Other	f282.2
2	Freshman	f282.4
3	Sophomore	f282.4
4	Sophomore	f282.1
5	Freshman	f282.3
6	Sophomore	f282.3
7	Junior	f282.1
8	Freshman	f282.4
9	Sophomore	f290.4
10	Sophomore	f290.4

In building the user dimension in SSIS, we needed to attend to two issues besides connecting the source to the destination. Specifically, we transformed the userId variable to integer, and the floorCode and yearInSchool to unicode strings. These processes were necessary for the subsequent merging and creation of the fact table.



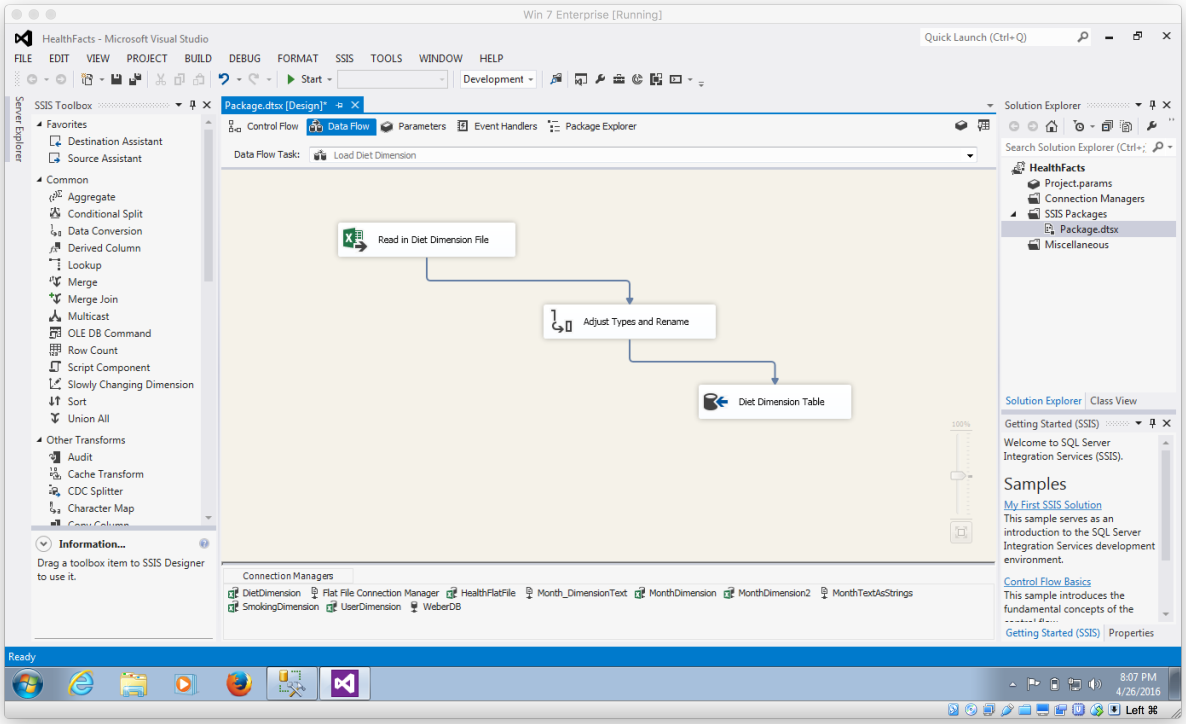
2.4.2 Diet Dimension

The diet dimension required similar proceesses as the construction of the user dimension. The dimension is rather simple, entailing only a diet status (present in the raw data) and a dietStatusId. The purpose of the Id is to facilitate analysis in the fact table use numerical levels rather than character levels. Moreover, it allows us to demonstrate that many other aspects could be included in the dimension table, should that need arise. We again used a data transformation to make the unicode strings agreeable during the construction of the fact table.

```r
library(RODBC)
diet_file <- sqlQuery(db_edweber, "SELECT *
                                   FROM   dim_diet_status")
knitr::kable(diet_file,
             digits = 2,
             caption = "Diet Dimension Table")
```

Table 3: Diet Dimension Table

dietStatusId	dietDescription
1	Very unhealthy
2	Unhealthy
3	Below average
4	Average
5	Healthy
6	Very healthy



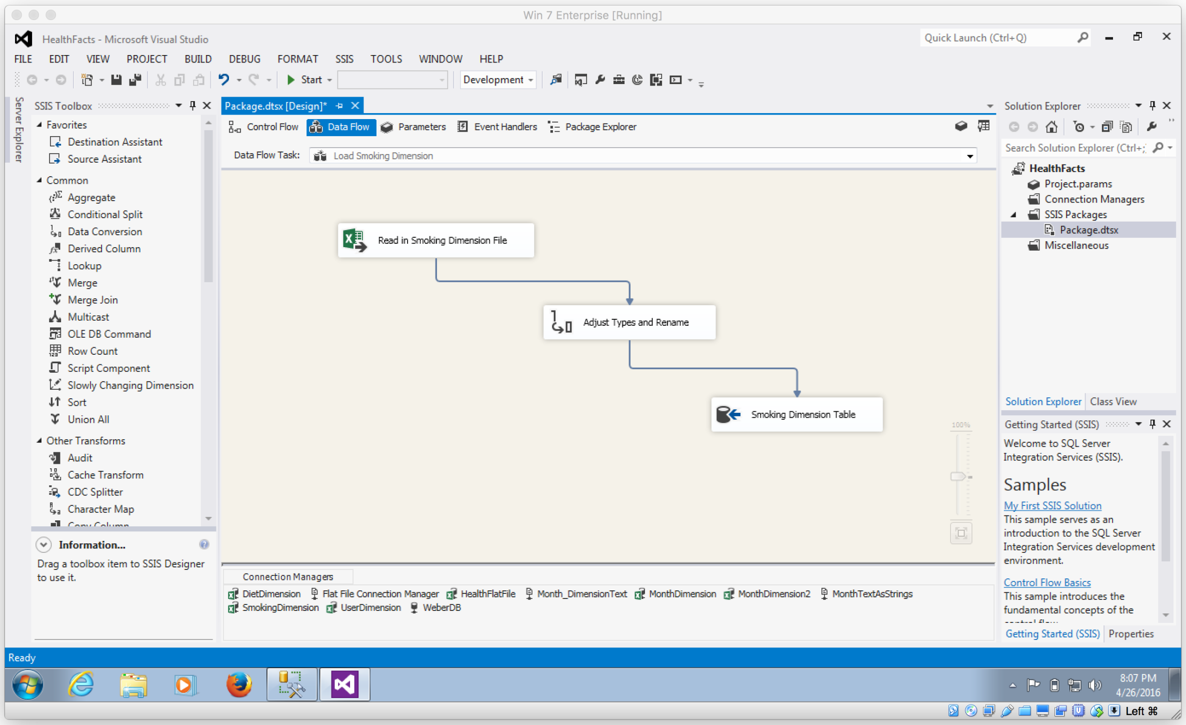
2.4.3 Smoking Dimension

The smoking dimension followed the same pattern as the diet dimension. The dimension includes the smokingStatus which is a text description comprising factor levels. The dimension table provides numerical identifiers for those levels, which can ease the analysis process or more advanced construction of models. It should be noted that to prepare for more advanced analysis, a large number of indicator variables would be appropriate. However, that would likely be done within statistical software, and not within the fact table itself.

```r
library(RODBC)
smoking_file <- sqlQuery(db_edweber, "SELECT *
                                      FROM   dim_smoking_status")
knitr::kable(smoking_file,
             digits = 2,
             caption = "Smoking Dimension Table")
```

Table 4: Smoking Dimension Table

smokingStatusId	smokingStatusDescription
1	Never
2	Once in a while
3	Some days
4	Every day

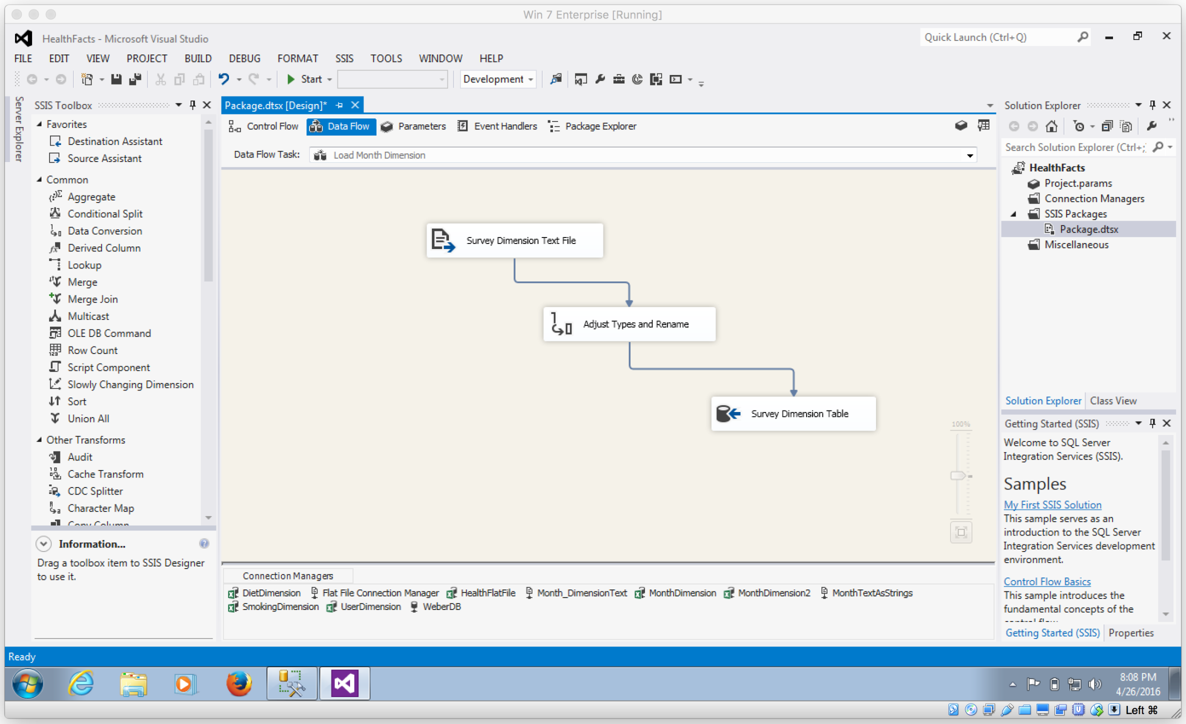


2.4.4 Time Dimension

The time dimension (specifically month and year of the survey) was the most complicated of the dimensions. The raw data provides dates in the format “yyyy.mm”, whic proved difficult for SSIS to parse. Even when read in as a string, at each step it converted the data to a double precision float value. Instead, we changed the dimension so that the primary key of the dimension table is in the format “yyyymm” with an associated date format. Of course, this requires manipulating the string data read in through SSIS prior to creating the fact table, but this appeared to us to be the most efficient way to overcome these string issues.

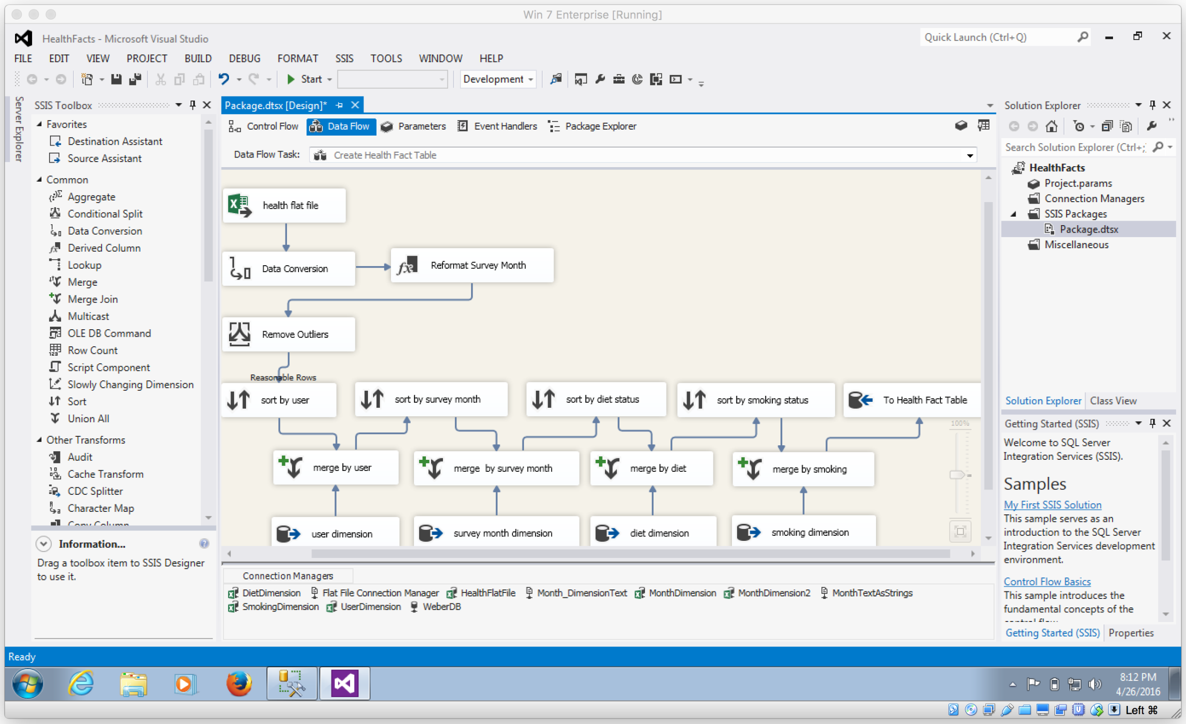
Table 5: Time Dimension Table

surveyMonthId	surveyMonthDescription
2008-08	200808
2008-09	200809
2008-10	200810
2008-11	200811
2008-12	200812
2009-01	200901
2009-02	200902
2009-03	200903
2009-04	200904
2009-05	200905
2009-06	200906



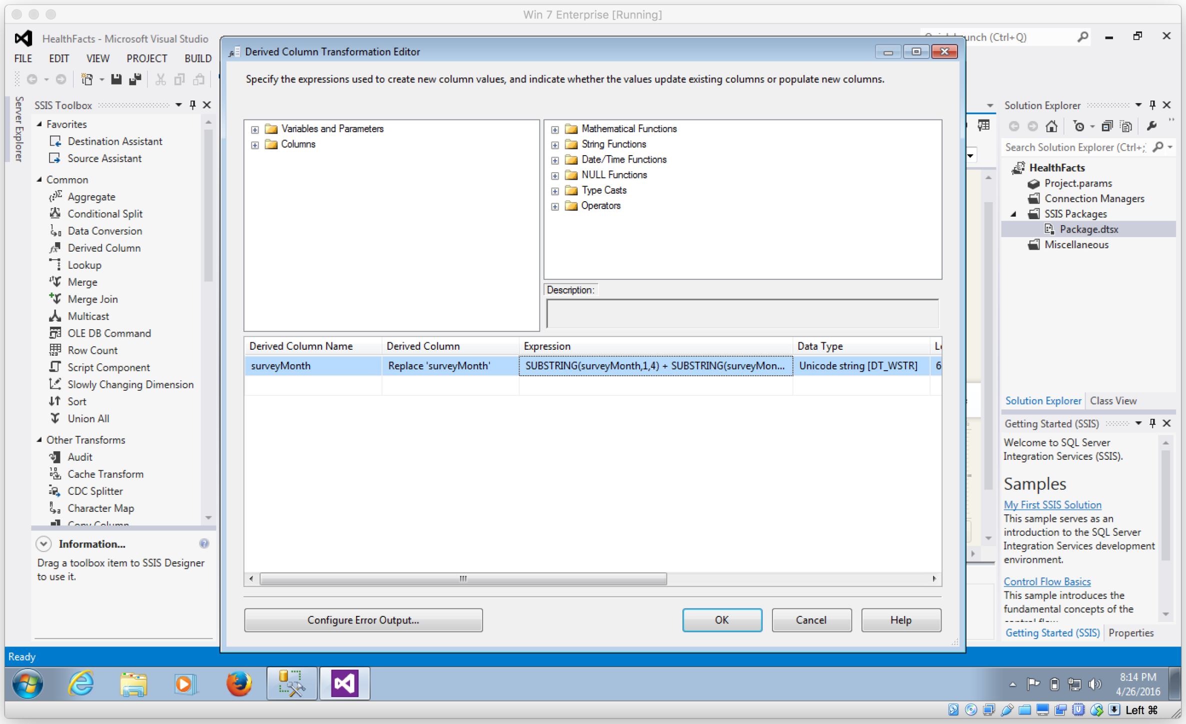
2.5 Fact Table Flow Description

The second container of the data flow contains a single flow, construction of the fact table. The overview of the process is in the figure below. The important steps include reading in the data from an excel file, transformating the data (adjusting strings, changing data types, filtering out unusual values), and a join for each dimension. We examine these processes in more detail in the figures below.



2.5.1 Important Modifications

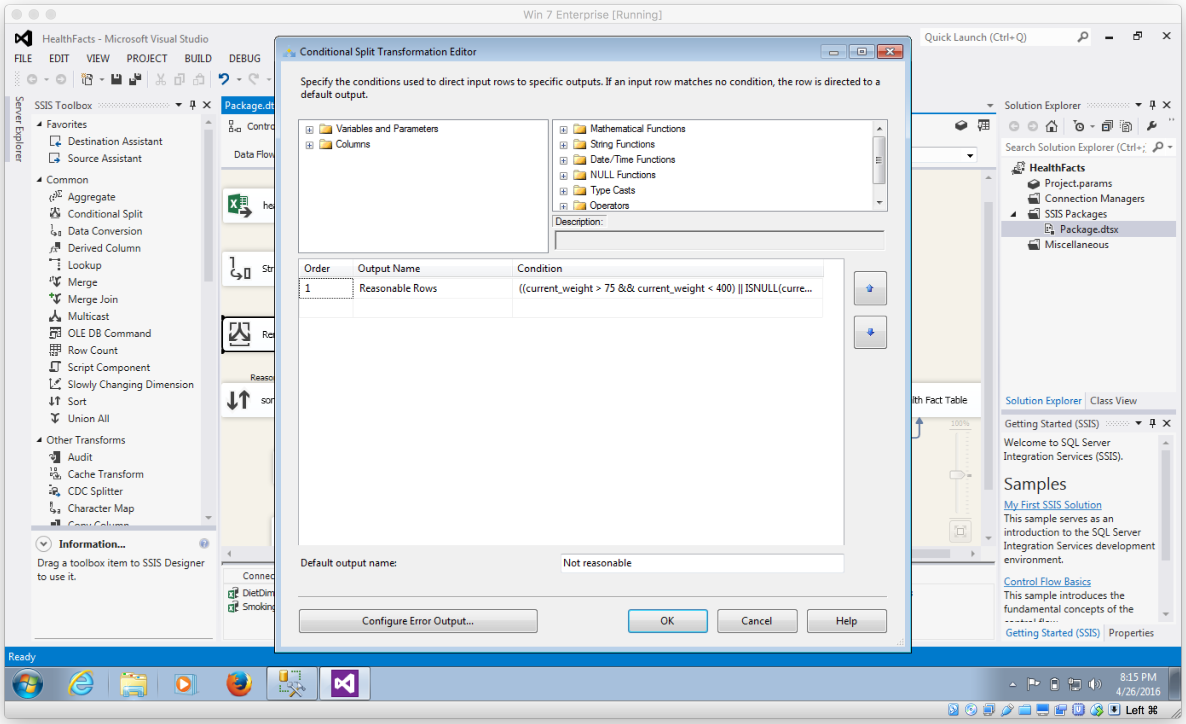
With the raw data as a source, we modified the time column (surveyMonth) to fit with the dimension table. The surveyMonth column comes in the form "yyyy.mm". The string manipulation grabs the substrings before and after the period and joins them, which allows for the join by the time dimension in subsequent steps.



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type	L
surveyMonth	Replace 'surveyMonth'	SUBSTRING(surveyMonth,1,4) + SUBSTRING(surveyMon...	Unicode string [DT_WSTR]	6

An initial inspection of the raw data shows there are a number of strange values present. Specifically, height values of over 2000 inches and weight values of 0. The next step filters the rows containing these highly unusual observations. We determined that while it is possible to change these values to NULL instead of filtering them out of the dataset, the observations themselves are suspect. In a more formal setting, or in analysis software like R or Python, we could quantify the effect of removing these observations. However, for our purposes here, we remove those rows from the fact table.



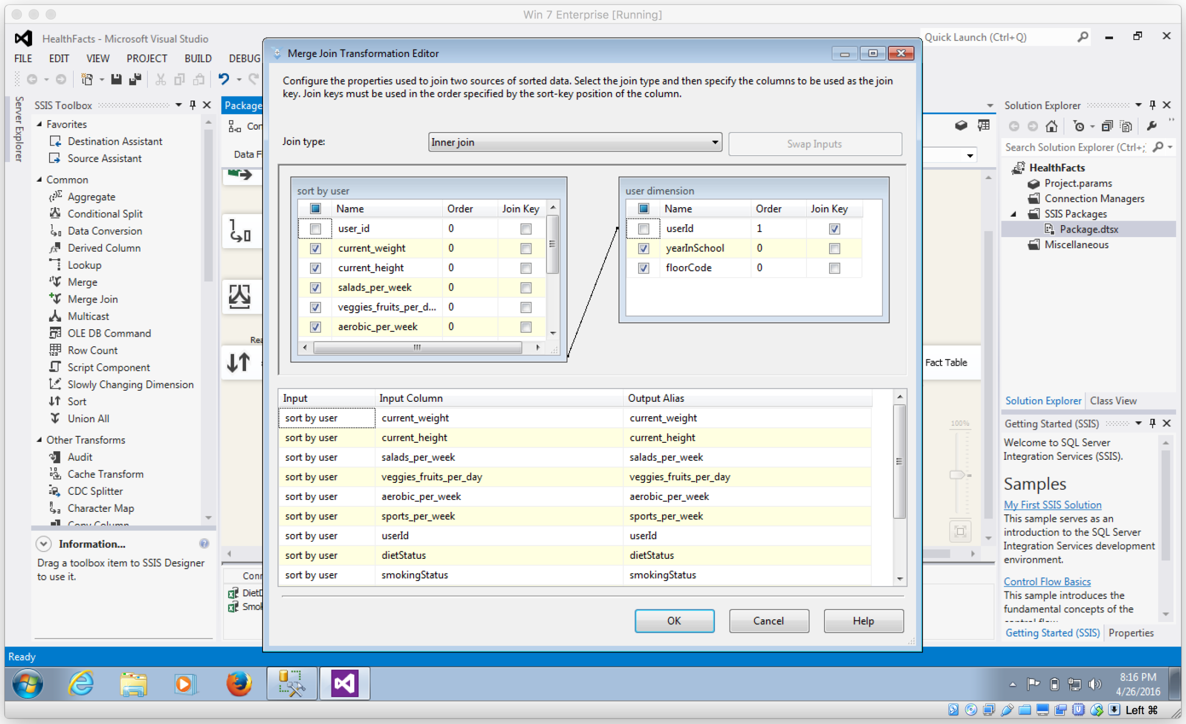
Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

Order	Output Name	Condition
1	Reasonable Rows	((current_weight > 75 && current_weight < 400) || ISNULL(curre...

Default output name: Not reasonable

2.5.2 Joins

The following four figures display the joins made to each of the dimensions for the fact table. Each join required sorting by the primary key of that dimension. In the first case, we used an inner join for users. This ensured that a user with an Id not in the dimension table would not be in the fact table. For the subsequent joins, we used left outer joins. We did this to ensure that even if the raw data row did not have a particular dimension, that observation was still present in the fact table.


Win 7 Enterprise [Running]
HealthFacts - Microsoft Visual Studio
Merge Join Transformation Editor
SSIS Toolbox
Solution Explorer
OK
Cancel
Help
Swap Inputs

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Inner join

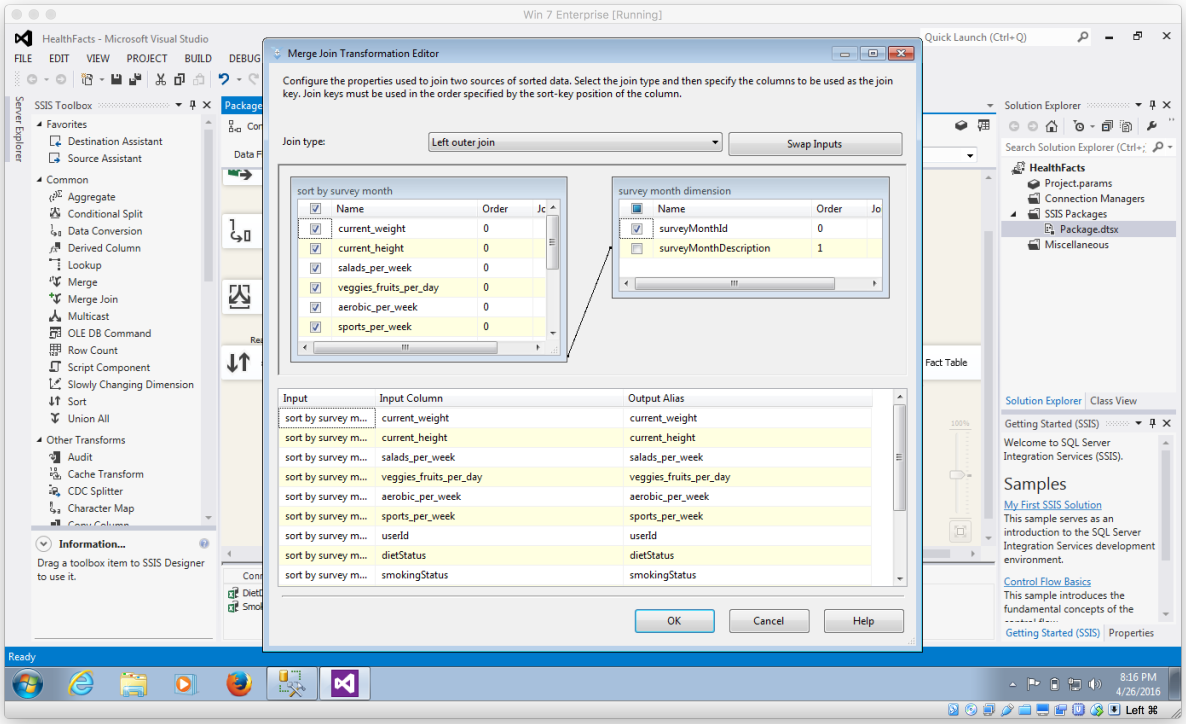
sort by user

	Name	Order	Join Key
☐	user_id	0	☐
✓	current_weight	0	☐
✓	current_height	0	☐
✓	salads_per_week	0	☐
✓	veggies_fruits_per_d...	0	☐
✓	aerobic_per_week	0	☐

user dimension

	Name	Order	Join Key
☐	userId	1	✓
✓	yearInSchool	0	☐
✓	floorCode	0	☐

Input	Input Column	Output Alias
sort by user	current_weight	current_weight
sort by user	current_height	current_height
sort by user	salads_per_week	salads_per_week
sort by user	veggies_fruits_per_day	veggies_fruits_per_day
sort by user	aerobic_per_week	aerobic_per_week
sort by user	sports_per_week	sports_per_week
sort by user	userId	userId
sort by user	dietStatus	dietStatus
sort by user	smokingStatus	smokingStatus


Win 7 Enterprise [Running]
HealthFacts - Microsoft Visual Studio
Merge Join Transformation Editor
SSIS Toolbox
Solution Explorer
OK
Cancel
Help
Swap Inputs

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Left outer join

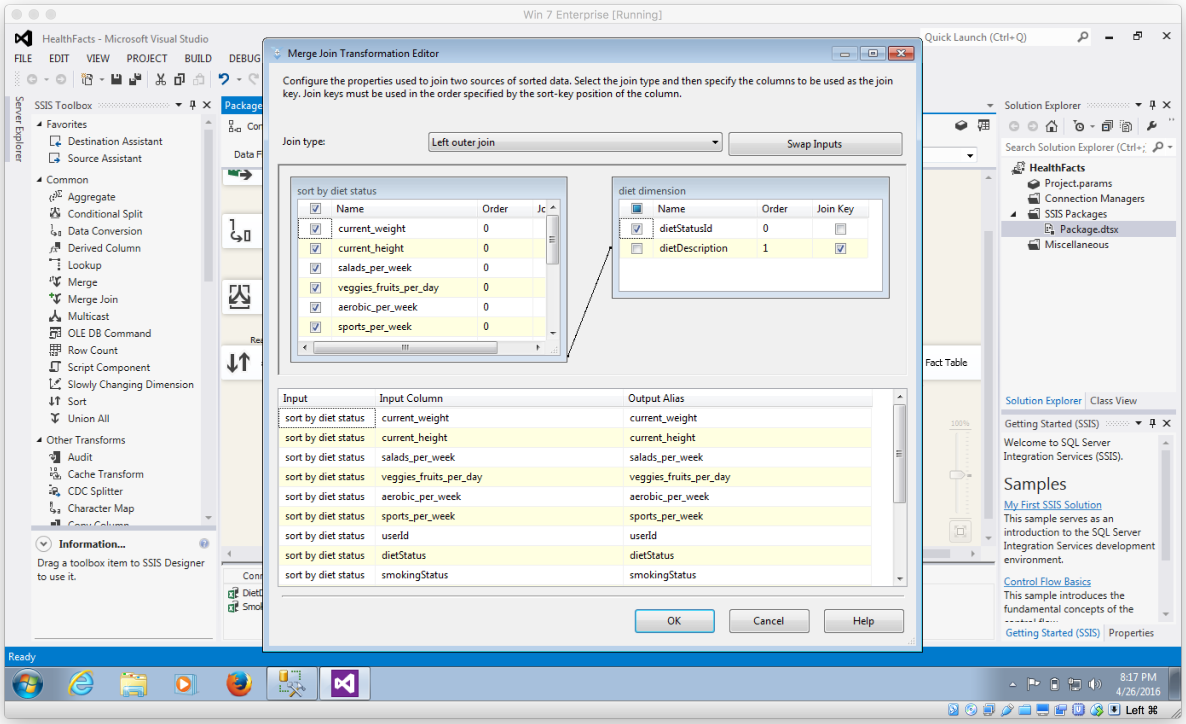
sort by survey month

	Name	Order	Jc
✓	current_weight	0	
✓	current_height	0	
✓	salads_per_week	0	
✓	veggies_fruits_per_day	0	
✓	aerobic_per_week	0	
✓	sports_per_week	0	

survey month dimension

	Name	Order	Jo
✓	surveyMonthId	0	
☐	surveyMonthDescription	1	

Input	Input Column	Output Alias
sort by survey m...	current_weight	current_weight
sort by survey m...	current_height	current_height
sort by survey m...	salads_per_week	salads_per_week
sort by survey m...	veggies_fruits_per_day	veggies_fruits_per_day
sort by survey m...	aerobic_per_week	aerobic_per_week
sort by survey m...	sports_per_week	sports_per_week
sort by survey m...	userId	userId
sort by survey m...	dietStatus	dietStatus
sort by survey m...	smokingStatus	smokingStatus


Win 7 Enterprise [Running]
HealthFacts - Microsoft Visual Studio
Merge Join Transformation Editor
SSIS Toolbox
Solution Explorer
OK
Cancel
Help
Swap Inputs
Ready

Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Left outer join

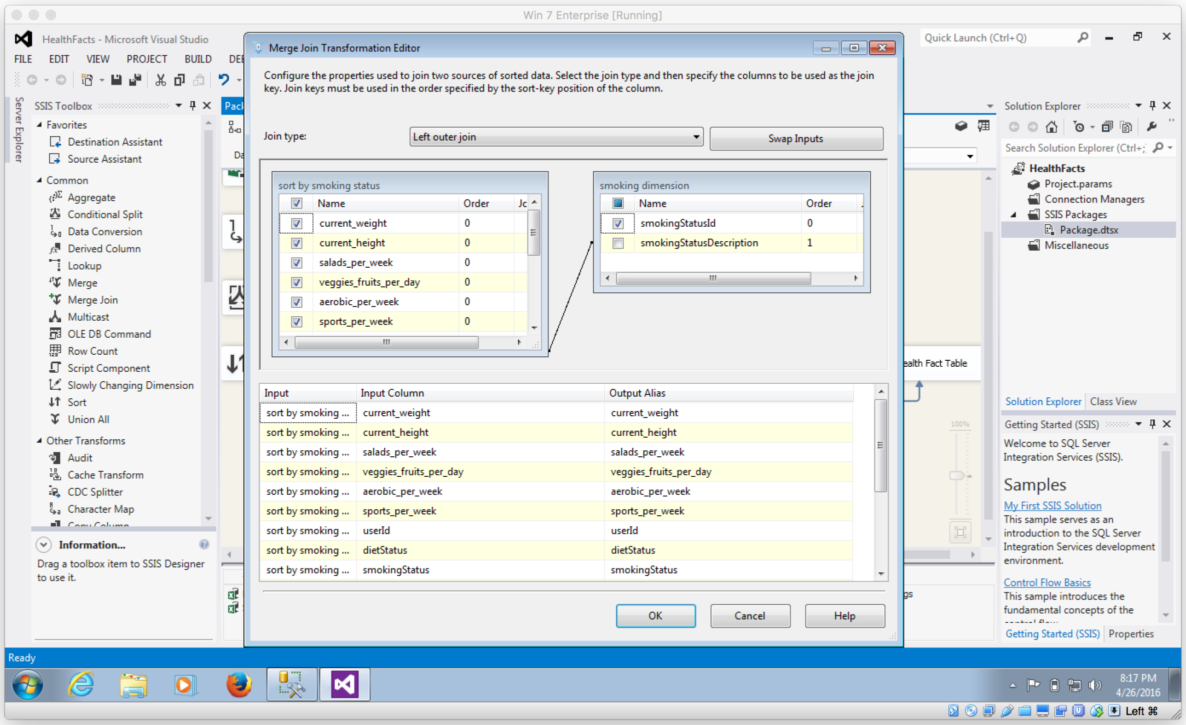
sort by diet status

	Name	Order
✓	current_weight	0
✓	current_height	0
✓	salads_per_week	0
✓	veggies_fruits_per_day	0
✓	aerobic_per_week	0
✓	sports_per_week	0

diet dimension

	Name	Order	Join Key
✓	dietStatusId	0	
	dietDescription	1	✓

Input	Input Column	Output Alias
sort by diet status	current_weight	current_weight
sort by diet status	current_height	current_height
sort by diet status	salads_per_week	salads_per_week
sort by diet status	veggies_fruits_per_day	veggies_fruits_per_day
sort by diet status	aerobic_per_week	aerobic_per_week
sort by diet status	sports_per_week	sports_per_week
sort by diet status	userId	userId
sort by diet status	dietStatus	dietStatus
sort by diet status	smokingStatus	smokingStatus


Win 7 Enterprise [Running]
HealthFacts - Microsoft Visual Studio
Merge Join Transformation Editor
SSIS Toolbox
Solution Explorer
OK
Cancel
Help
Swap Inputs
Ready

Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Left outer join

sort by smoking status

	Name	Order
✓	current_weight	0
✓	current_height	0
✓	salads_per_week	0
✓	veggies_fruits_per_day	0
✓	aerobic_per_week	0
✓	sports_per_week	0

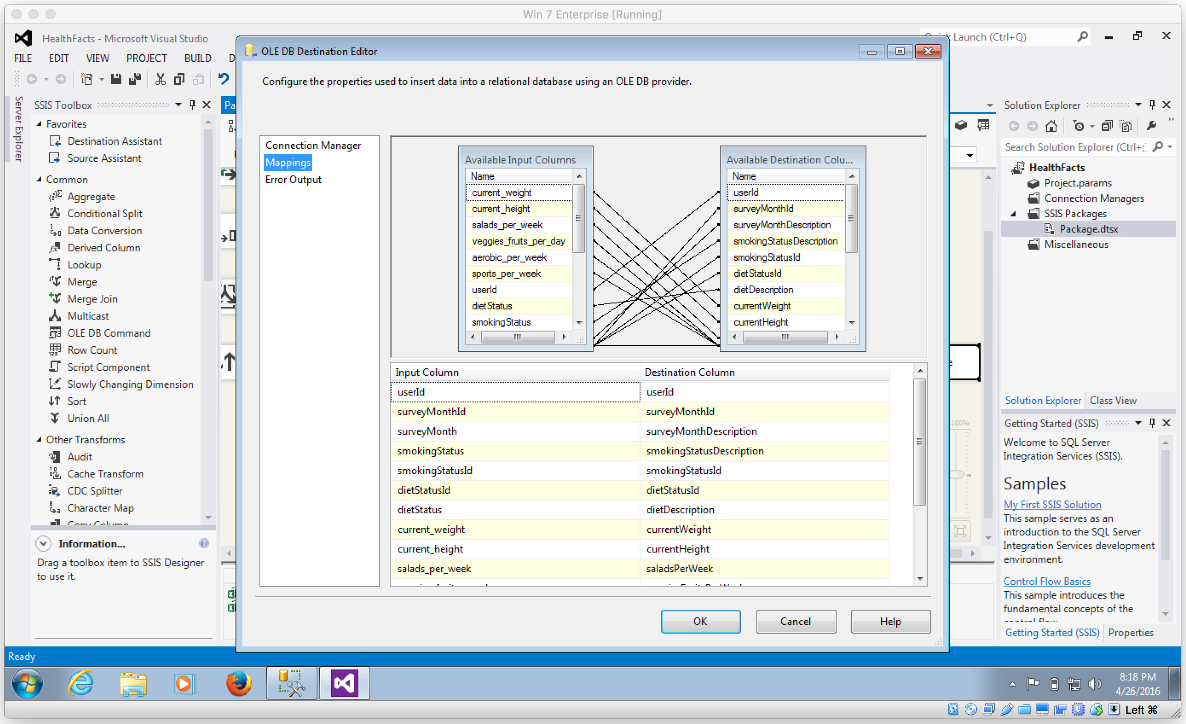
smoking dimension

	Name	Order
✓	smokingStatusId	0
	smokingStatusDescription	1

Input	Input Column	Output Alias
sort by smoking ...	current_weight	current_weight
sort by smoking ...	current_height	current_height
sort by smoking ...	salads_per_week	salads_per_week
sort by smoking ...	veggies_fruits_per_day	veggies_fruits_per_day
sort by smoking ...	aerobic_per_week	aerobic_per_week
sort by smoking ...	sports_per_week	sports_per_week
sort by smoking ...	userId	userId
sort by smoking ...	dietStatus	dietStatus
sort by smoking ...	smokingStatus	smokingStatus

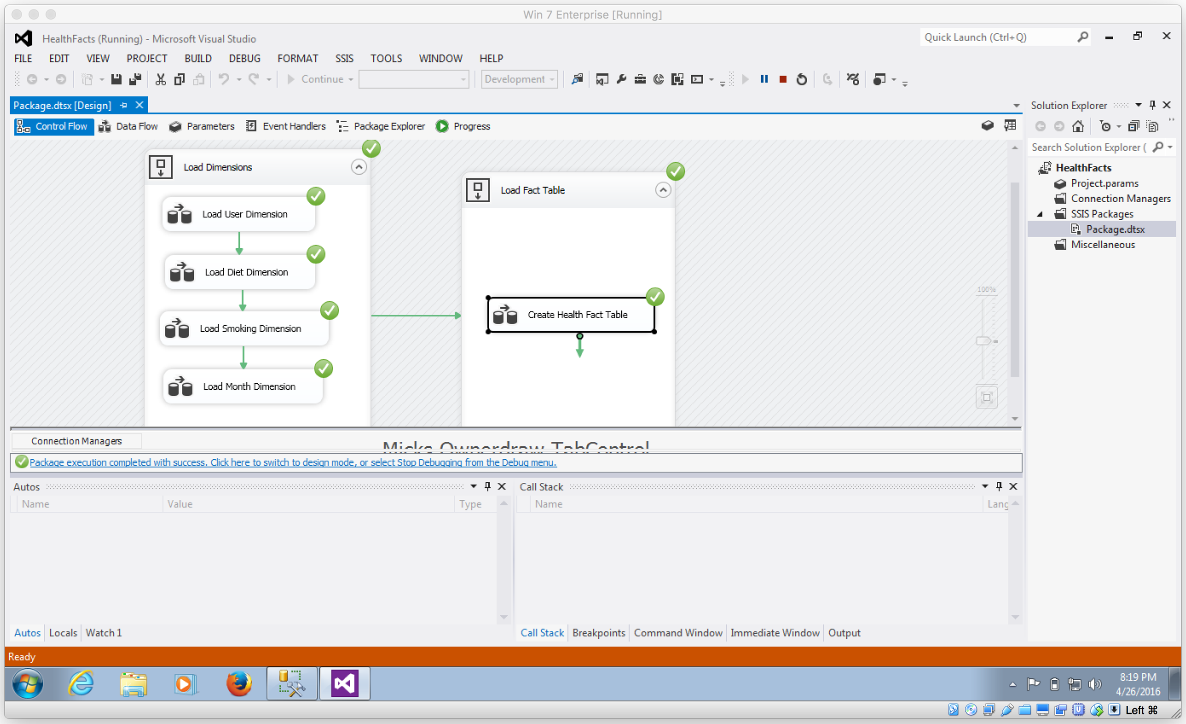
2.5.3 Construction of Fact Table

Once the joins were completed, we matched the columns resulting from the joins with the columns of the fact table present on the SQL Server. The next figure shows the data flows executed succesfully.


Win 7 Enterprise [Running]
HealthFacts - Microsoft Visual Studio
OLE DB Destination Editor
Connection Manager
Mappings
Error Output
Available Input Columns
Available Destination Columns
SSIS Toolbox
Solution Explorer
OK
Cancel
Help
Ready

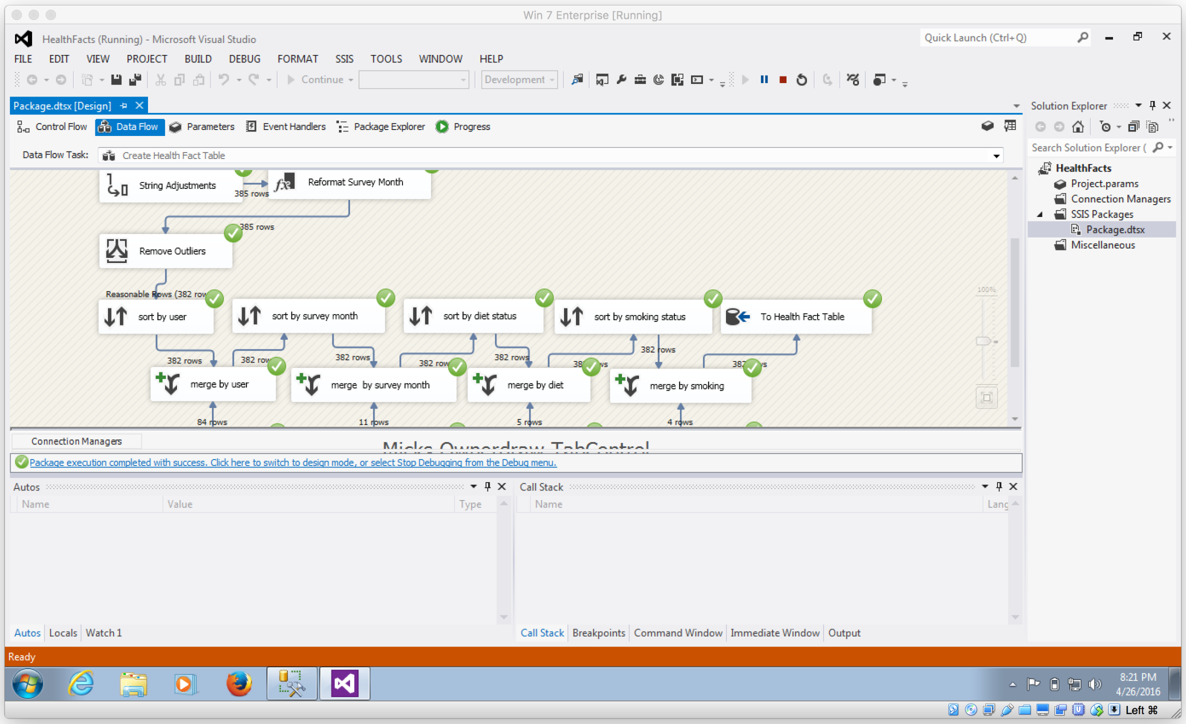
Configure the properties used to insert data into a relational database using an OLE DB provider.

Input Column	Destination Column
userId	userId
surveyMonthId	surveyMonthId
surveyMonth	surveyMonthDescription
smokingStatus	smokingStatusDescription
smokingStatusId	smokingStatusId
dietStatusId	dietStatusId
dietStatus	dietDescription
current_weight	currentWeight
current_height	currentHeight
salads_per_week	saladsPerWeek

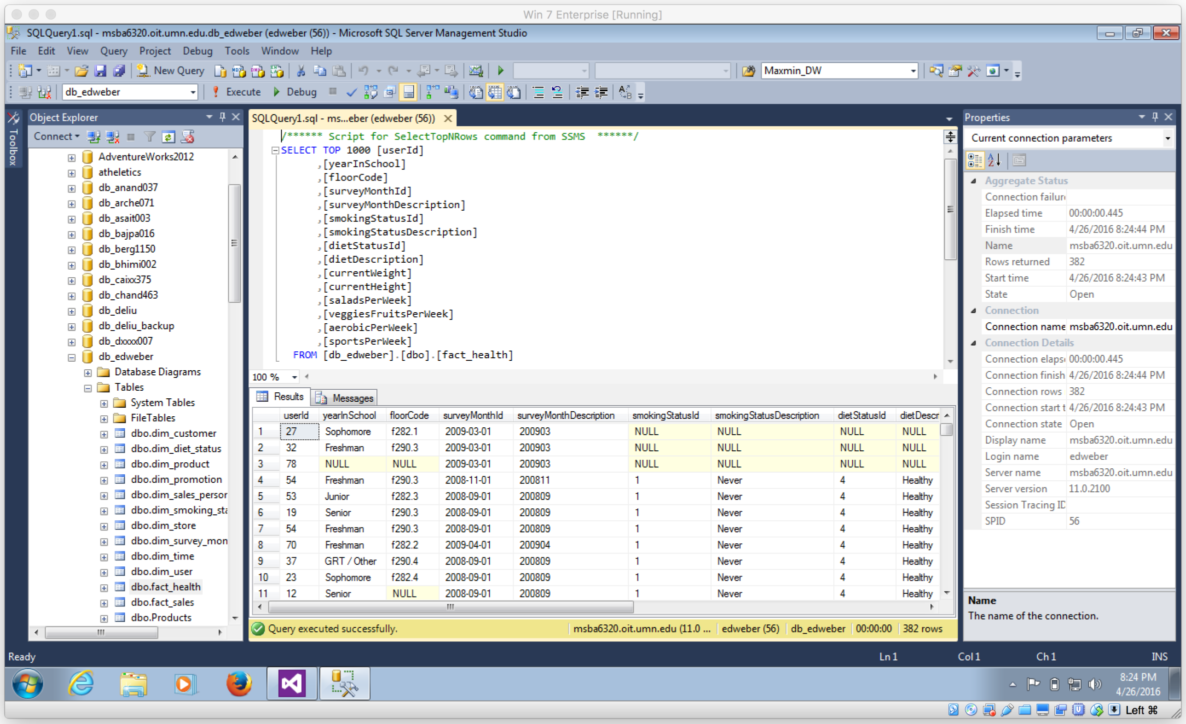


Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

The figure below shows how many rows were present at each step of the construction of the fact table, which matches with the number of rows in the fact table on the SQL Server.



Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.



```sql
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [userId]
      ,[yearInSchool]
      ,[floorCode]
      ,[surveyMonthId]
      ,[surveyMonthDescription]
      ,[smokingStatusId]
      ,[smokingStatusDescription]
      ,[dietStatusId]
      ,[dietDescription]
      ,[currentWeight]
      ,[currentHeight]
      ,[saladsPerWeek]
      ,[veggiesFruitsPerWeek]
      ,[aerobicPerWeek]
      ,[sportsPerWeek]
  FROM [db_edweber].[dbo].[fact_health]
```

	userId	yearInSchool	floorCode	surveyMonthId	surveyMonthDescription	smokingStatusId	smokingStatusDescription	dietStatusId	dietDescr
1	27	Sophomore	f282.1	2009-03-01	200903	NULL	NULL	NULL	NULL
2	32	Freshman	f290.3	2009-03-01	200903	NULL	NULL	NULL	NULL
3	78	NULL	NULL	2009-03-01	200903	NULL	NULL	NULL	NULL
4	54	Freshman	f290.3	2008-11-01	200811	1	Never	4	Healthy
5	53	Junior	f282.3	2008-09-01	200809	1	Never	4	Healthy
6	19	Senior	f290.3	2008-09-01	200809	1	Never	4	Healthy
7	54	Freshman	f290.3	2008-09-01	200809	1	Never	4	Healthy
8	70	Freshman	f282.2	2009-04-01	200904	1	Never	4	Healthy
9	37	GRT / Other	f290.4	2008-09-01	200809	1	Never	4	Healthy
10	23	Sophomore	f282.4	2008-09-01	200809	1	Never	4	Healthy
11	12	Senior	NULL	2008-09-01	200809	1	Never	4	Healthy

Query executed successfully. | msba6320.oit.umn.edu (11.0 ... | edweber (56) | db_edweber | 00:00:00 | 382 rows

Properties: Current connection parameters

Aggregate Status	
Connection failur	
Elapsed time	00:00:00.445
Finish time	4/26/2016 8:24:44 PM
Name	msba6320.oit.umn.edu
Rows returned	382
Start time	4/26/2016 8:24:43 PM
State	Open
Connection	
Connection name	msba6320.oit.umn.edu
Connection Details	
Connection elaps	00:00:00.445
Connection finish	4/26/2016 8:24:44 PM
Connection rows	382
Connection start t	4/26/2016 8:24:43 PM
Connection state	Open
Display name	msba6320.oit.umn.edu
Login name	edweber
Server name	msba6320.oit.umn.edu
Server version	11.0.2100
Session Tracing ID	
SPID	56

Name
The name of the connection.

3 Analysis and Results

This part of the project focused on creating a connection between the fact table and R for analysis purposes. The RODBC library, and some additional server mapping adjustments, allow for querying from the database without bringing all of the data into R. For certain data visualization, bringing aspects of the data into the analysis suite is useful, but in a truly large dataset, it is much easier to query, particularly in a program that uses local memory like R.

3.1 Issue 1: Survey Respondents

It is important to consider the characteristics of the users who reported, as bias in reporting can certainly skew the data. We used two queries, one to generate the number of reports by floor, and one to generate the number of users on each floor (from the dimension table). These two tables allowed for creation of an average number of reports per student on each floor. The table below illustrates that those users without a given floor code generated fewer average reports, and that three floors averaged 5.8 to 6.0 reports per student.

```
report_by_floor <- sqlQuery(db_edweber, "SELECT  floorCode,
                                                 COUNT(*) AS totalReportsFloor
                                        FROM     fact_health
                                        GROUP BY floorCode")

users_on_floor <- sqlQuery(db_edweber, "SELECT floorCode,
                                               COUNT(*) AS totalUsersFloor
                                        FROM   dim_user
                                        GROUP BY floorCode")

combined_users <- cbind(report_by_floor,
                        users_on_floor)[, - c(3)]

combined_users$avgReportsPerUser <- combined_users$totalReportsFloor / +
    combined_users$totalUsersFloor

knitr::kable(combined_users,
             digits = 2,
             caption = "Reports by Floor")
```

Table 6: Reports by Floor

floorCode	totalReportsFloor	totalUsersFloor	avgReportsPerUser
NA	22	10	2.20
f282.1	58	10	5.80
f282.2	44	9	4.89
f282.3	70	15	4.67
f282.4	54	14	3.86
f290.1	6	1	6.00
f290.2	26	5	5.20
f290.3	64	12	5.33
f290.4	38	8	4.75

Given that the MIT dorms are not freshmen only, we examined if the average report rate per user depended on the student's year in school, as provided by the dimension table. As above, we created a table that allowed us to measure average reports per user for each year in school. While the rates are slightly different, the differences are not meaningful enough to suggest substantial bias produced by a single class of student. We examined a number of other aspects of the data to examine influence of a particular group on the results, but did not find any evidence of that in the data provided.

```
report_by_year <- sqlQuery(db_edweber, "SELECT   yearInSchool,
                                                 COUNT(*) AS totalReportsYear
                                        FROM     fact_health
                                        GROUP BY yearInSchool")

users_by_year <- sqlQuery(db_edweber, "SELECT  yearInSchool,
                                               COUNT(*) AS totalUsersYear
                                        FROM   dim_user
                                        GROUP BY yearInSchool")

combined_year <- cbind(report_by_year,
                       users_by_year)[, - c(3)]

combined_year$avgReportsPerUser <- combined_year$totalReportsYear / +
    combined_year$totalUsersYear

knitr::kable(combined_year,
             digits = 2,
             caption = "Reports by Year in School")
```

Table 7: Reports by Year in School

yearInSchool	totalReportsYear	totalUsersYear	avgReportsPerUser
NA	17	11	1.55
Freshman	109	21	5.19
GRT / Other	42	9	4.67
Junior	35	8	4.38
Senior	70	14	5.00
Sophomore	109	21	5.19

3.2 Issue 2: Health by Floor

Once we concluded that a single floor or single class was not overly represented in the data, we considered how health outcomes might vary across dimensions of the data. As above, we focus on floor and year in school for our analyses, while noting that there are a large number of other possible avenues to pursue. We use these analyses as demonstrations of what more could be done with the data, not as an exhaustive analysis of it.

The table below examines averages of each of the health variables across floor. It appears that some floors certainly have higher average weights and heights than other floors. Similarly, some floors have healthier eating habits than do other floors. One issue to consider is the relatively small number of reports that came from f290.1.

```
measures_by_floor <- sqlQuery(db_edweber, "SELECT   floorCode,
                                                    COUNT(*) AS totalReports,
                                                    AVG(currentWeight) AS avgWt,
                                                    AVG(currentHeight) AS avgHt,
                                                    AVG(saladsPerWeek) AS avgSalads,
                                                    AVG(veggiesFruitsPerWeek) AS avgVF,
                                                    AVG(aerobicPerWeek) AS avgActivity
                                            FROM     fact_health GROUP BY floorCode")
knitr::kable(measures_by_floor,
             digits = 2,
             caption = "Student Health Measures by Floor")
```

Table 8: Student Health Measures by Floor

floorCode	totalReports	avgWt	avgHt	avgSalads	avgVF	avgActivity
NA	22	147	64	0	2	1
f282.1	58	205	69	1	1	1
f282.2	44	143	64	2	3	2
f282.3	70	148	69	1	2	2
f282.4	54	161	66	0	1	1
f290.1	6	129	65	2	2	2
f290.2	26	162	67	0	1	1
f290.3	64	130	64	1	2	2
f290.4	38	165	70	1	2	3

3.3 Issue 3: Health by School Status

We also examined variation in health measures by year in school (or school status). We found, as expected, that the maority of observations came from freshmen and sophomore students, which makes sense given we examined students who lived on campus. While there certainly seem to be some differences in terms of weight and average height (an ANOVA test does confirm this, but is beyond the scope of our discussion here). These measurements also raise the issue of self reports. There is no guarantee all the students thought about servings of vegetables, or aerobic activities, in the same way. Real time data, from fitbit, phone accelerometers, and apps like myfitnesspal would prove much more useful for distinguishing and measuring true differences in these groups.

```
measures_by_year <- sqlQuery(db_edweber, "SELECT    yearInSchool,
                                                    COUNT(*) AS totalObservations,
                                                    AVG(currentWeight) AS avgWt,
                                                    AVG(currentHeight) AS avgHt,
                                                    AVG(saladsPerWeek) AS avgSalads,
                                                    AVG(veggiesFruitsPerWeek) AS avgVF,
                                                    AVG(aerobicPerWeek) AS avgActivity
                                            FROM     fact_health
                                            GROUP BY yearInSchool")
knitr::kable(measures_by_year,
             digits = 2,
             caption = "Student Health Measures by Year in School")
```

Table 9: Student Health Measures by Year in School

yearInSchool	totalObservations	avgWt	avgHt	avgSalads	avgVF	avgActivity
NA	17	155	66	0	2	1
Freshman	109	145	66	1	2	2
GRT / Other	42	166	69	1	2	2

yearInSchool	totalObservations	avgWt	avgHt	avgSalads	avgVF	avgActivity
Junior	35	174	66	1	1	1
Senior	70	152	65	0	1	1
Sophomore	109	163	68	1	1	2

3.4 Issue 4: Health Over Time

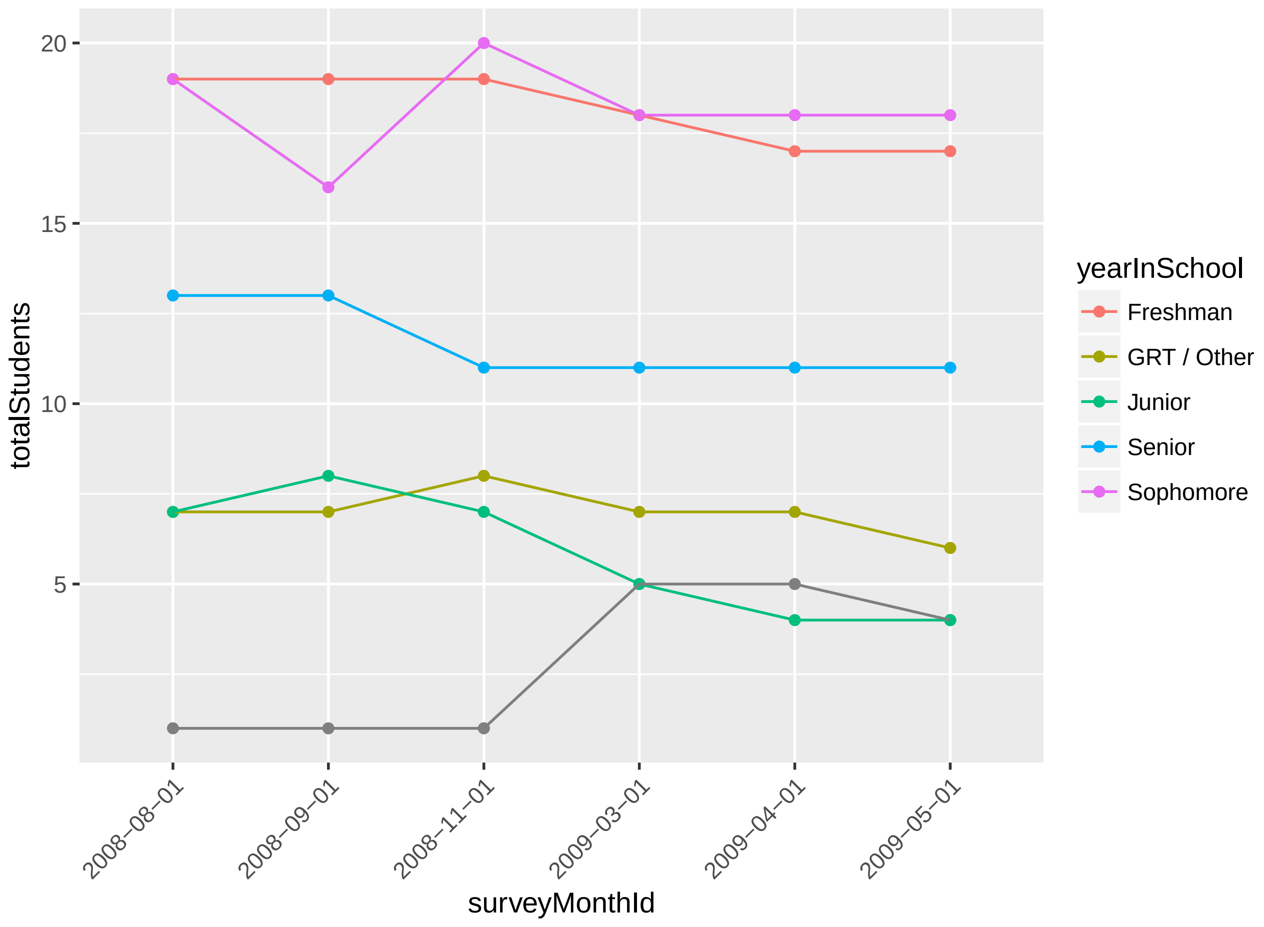
```
measures_by_time <- sqlQuery(db_edweber, "SELECT    surveyMonthId,
                                                    yearInSchool,
                                                    COUNT(*) AS totalStudents,
                                                    AVG(currentWeight) AS avgWt,
                                                    AVG(currentHeight) AS avgHt,
                                                    AVG(saladsPerWeek) AS avgSalads,
                                                    AVG(veggiesFruitsPerWeek) AS avgVF,
                                                    AVG(aerobicPerWeek) AS avgActivity
                                           FROM     fact_health
                                           GROUP BY surveyMonthId,
                                                    yearInSchool")
```

A final issue we focused on students as aggregate were any changes in health by survey month and by year in school. The data tables for this analysis were quite large and difficult to examine. Thus, we provided three graphs below that examined changes in health measures over time (by survey month, in this case). Of course, as with the other analyses, self reported measures should be met with some degree of skepticism.

3.4.1 Total Students for Survey Month by School Status

Prior to examining trends in health measures over time, it is critical to understand the characteristics of those who responded in each survey month. The plot below represents the number of students from each year in school to respond in the given month. Note that while the students could respond multiple times over the course of the year, they were able to only submit one response per month, so this count does represent distinct students. As before, it is clear the freshmen and sophomores, who are overrepresented in the dorm, also represent the largest proportion of responses in each month. Note the gray lines represent the students not attributed to a specific class..

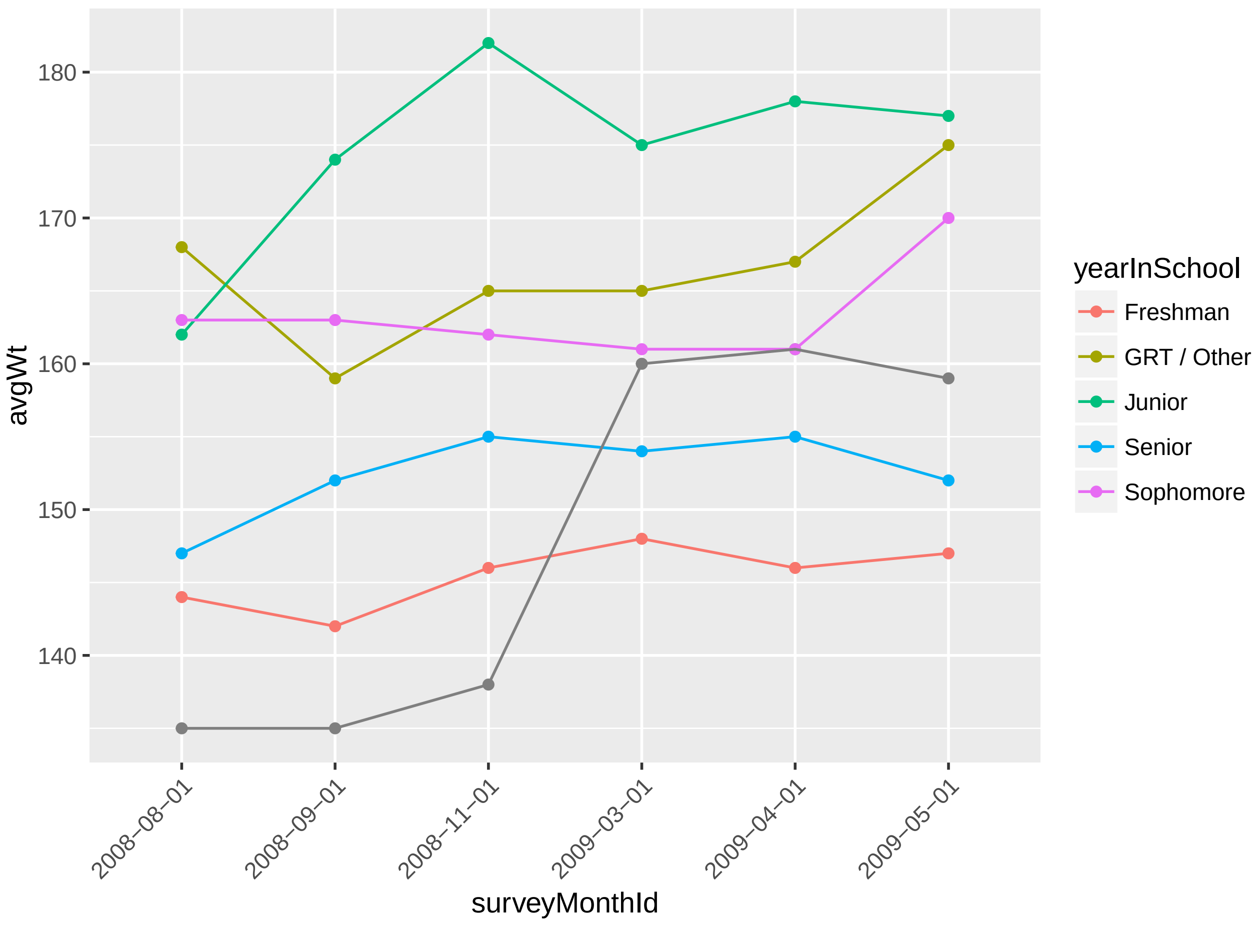
```
ggplot(measures_by_time, aes(x=surveyMonthId,
                             y=totalStudents,
                             color=yearInSchool,
                             group = yearInSchool)) +
                         geom_point() +
                         geom_line() +
                         theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



3.4.2 Average Weight vs Survey Month by School Status

The graph below represents the average weight of students by year in school over the course of the academic year. The gray line represents those responses that cannot be attributed to a specific class. It is clear that there is a difference between freshmen average weight and junior average weight, with the other classes falling largely in between. Looking back at the measures of height earlier in the report, the juniors tended to be taller and heavier, and also live in the same location in the dorm. This is a pattern that would need further investigation, but perhaps a group of friends who are also athletes (given the weights around 300 pounds) lived in that area.

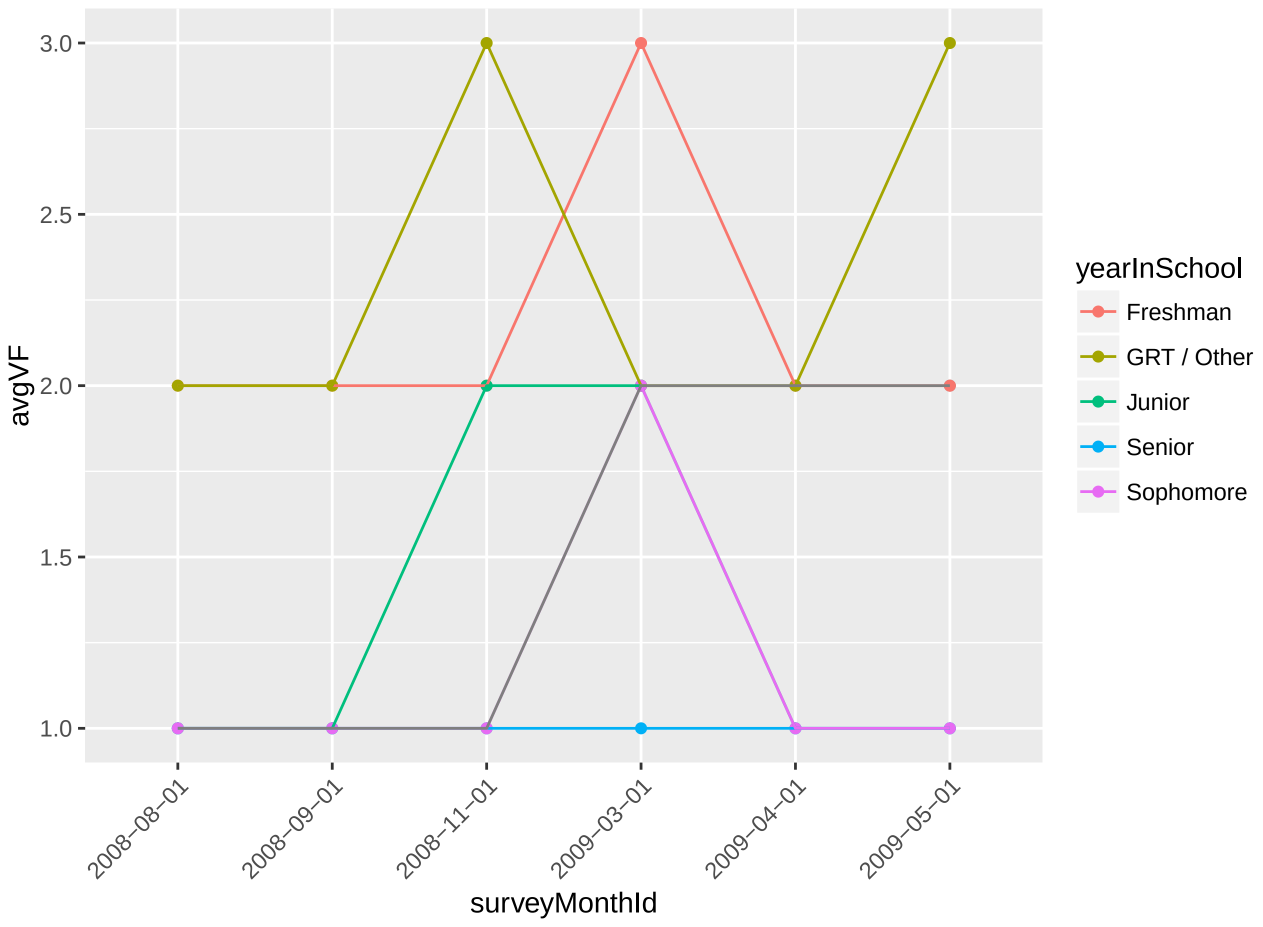
```
ggplot(measures_by_time, aes(x=surveyMonthId,
                             y=avgWt,
                             color=yearInSchool,
                             group = yearInSchool)) +
                         geom_point() +
                         geom_line() +
                         theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



3.4.3 Average Vegetables and Fruits vs Survey Month by School Status

The final graph we present focuses on vegetables and fruits consumed by school status. We include this not so much to illustrate an important difference, but to suggest that the data simply does not have enough variability, and likely not enough precision, to be useful in the way we intend. More specific measures, such as serving size, calories, frequency of eating, and the like, would be enormously useful to someone wanting to understand what help the students need to be healthy. For our own purposes of analysis, there simply is not enough variation in this particular variable for a deep analysis to be possible.

```r
ggplot(measures_by_time, aes(x=surveyMonthId,
                             y=avgVF,
                             color=yearInSchool,
                             group = yearInSchool)) +
                         geom_point() +
                         geom_line() +
                         theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



4 Conclusions

Conclusions:

4.1 The ETL Process

As anticipated, the ETL process was the most difficult and time-consuming aspect of this project. Specifically, a large portion of time was spent transforming the data into usable and workable formats. We had to change the data types of certain fields, such as userId and floorCode, in order to subsequently use them for merging tables later on. We also had to create and insert Ids for variables like diet status and smoking status so as to be able to use them as numerical rather than character variables in our analysis. The time dimension – with year and month data – also proved to be the most complicated data to clean and work with. The raw data was formatted in such a way that it made it difficult for SSIS to parse, and so we ended up having to change the format and also change the data type that the data was read in as. Finally, we had to parse out strange and outlier values like unrealistic height and weights so as to avoid erroneously skewing the data.

In addition to preparing our data for load, preparing the process to have the data load into our tables successfully was also very time-consuming. For each of the dimension tables, the data needed to be adjusted or re-formatted in some way before getting loaded into each of the tables. In many cases we needed to transform the data types, or remove erroneous values before loading into the tables. Once the dimension tables were created and data loaded, we began the process of joining them together with the fact table. This process proved to be somewhat tricky, as the data for each dimension needed to be sorted prior to merging with the next table. After some trial and error in figuring out the right type of join at each step, we were able to see it that all of the data flowed through to the end product – a fully functioning fact table!

The ETL portion of our project was definitely a learning process, in which we encountered several roadblocks and had to figure out how to navigate around various issues with the data. As we continue to build upon our data warehouse and bring in additional data sources this will likely continue to be the most time-consuming aspect of maintaining our data warehouse, as each new dataset will present its own defects and challenges. However, with the key structures in place this process should hopefully become smoother with each iteration.

4.2 Research Questions

4.2.1 Significance of Initial Analysis

Our analysis thus far has only scratched the surface of what we would like or be able to investigate using our data warehouse, but even our preliminary analysis has yielded some interesting results. We found some interesting patterns when looking at weight by year in school and floor; it appears as though juniors tended to be taller and heavier, and also live in the same dorm. To make this analysis more interesting, it would be nice if we could pull in relational and/or movement data to identify friendship groups and their movement patterns (or lack thereof).There was also a slight, but noticeable increase in weight across all groups over the course of the academic year – particularly amongst sophomores. We would like to be able to parse this out a bit more and identify the behavioral patterns of students who saw an increase in weight over the course of the year. We also took a preliminary look at the healthy eating habits of students in each class, specifically how many servings of fruits and vegetables the students ate on a regular basis. Given that our dataset is entirely self-reported, it was difficult to draw any sort of meaningful analysis from the data that we have currently. Our analysis showed no significant difference between the eating habits of each class, but without any sort of precise measures such as calorie count, service size, etc, it's difficult to conclude whether this analysis is truly meaningful.

4.2.2 Future Research

With more time, there are several aspects of our data warehouse and dataset that we would like to build out and investigate further. We would like to start layering in device data, both from cell phones and wearable devices such as Fitbit and the Apple watch. This data would be immensely useful in helping us identify relationships between students as well as their movement, activity, and behavioral patterns. This information would also add a level of precision that is lacking in our current dataset, in that we would not be privy to the sort of errors that can come out of self-reporting. We would also like to take our analysis to the next level and not just identify patterns in the data, but also use these patterns to help create predictive models that could help residence hall directors be proactive in identifying and responding to trends within the student population.

4.3 Future Requirements of Data for Analysis Purposes

In order to support the type and level of analysis that we would like to do going forward, there are several additional data requirements that we must layer on our existing model. As a first step, we would like to revisit the methods for capturing aspects of student health data – specifically how we can add in more precision and introduce as much standardization as possible in the responses. Some ways we might be able to do this would be by defining what we mean by serving size, asking for more specific metrics like calories, and/or even pulling this information from devices such as myfitnesspal. We would also like to build upon our current data warehouse and create additional fact table so as to support data from additional sources, such as:

- Cell phone and movement data (from current dataset)
- Students' interests and disease info (from current dataset)

- Real time data from fitbit, phone accelerometers, and apps like myfitnesspal
- Additional de-identified student info, such as course of study or ethnographic information

The data warehouse that we created thus far has served as an important stepping stone in building a resource for residence hall directors to gather and analyze information about their student population. By building on what we have so far and pulling in additional data sources, we will only continue to increase the value our tool provides and allow residence hall directors to gain meaningful insights.