Lisa Over
February 25, 2015
DW Assignment

Problem 1

Question 1

QUERY

```
SELECT cust.Gender, COUNT(*) as Purchases
FROM FactInternetSales as fact INNER JOIN
              DimProduct as prod ON fact.ProductKey = prod.ProductKey INNER JOIN
              DimProductSubcategory as subcat ON prod.ProductSubcategoryKey =
subcat.ProductSubcategoryKey INNER JOIN
              DimCustomer as cust ON fact.CustomerKey = cust.CustomerKey

WHERE subcat.EnglishProductSubcategoryName LIKE 'Jerseys'
GROUP BY cust.Gender
ORDER BY 1
```

RESULTS

Gender	Purchases
F	1652
M	1680

CONCLUSION

Men purchased more jersey's from the Internet, however, they purchased only 1.69% more than women.

Question 2

QUERY

```
SELECT dt.CalendarYear, cust.Gender, COUNT(*) as Purchases
FROM FactInternetSales as fact INNER JOIN
              DimProduct as prod ON fact.ProductKey = prod.ProductKey INNER JOIN
              DimProductSubcategory as subcat ON prod.ProductSubcategoryKey =
subcat.ProductSubcategoryKey INNER JOIN
              DimCustomer as cust ON fact.CustomerKey = cust.CustomerKey INNER JOIN
              DimDate as dt ON fact.OrderDateKey = dt.DateKey

WHERE subcat.EnglishProductSubcategoryName LIKE 'Jerseys'
GROUP BY dt.CalendarYear, cust.Gender
ORDER BY 1, 2 DESC
```

RESULTS

CalendarYear	Gender	Purchases
2007	M	668
2007	F	686
2008	M	1012
2008	F	966

Lisa Over
February 25, 2015
DW Assignment

CONCLUSION

In 2007, 668 jersey's were sold to men and 686 were sold to women. In 2008, 1012 jersey's were sold to men and 966 were sold to men. Sales to men increased by 44% while sales to women increased by 29% between 2007 and 2008.

Problem 2

QUERY

```
USE [AdventureWorksDW2012]
GO

SELECT promo.EnglishPromotionName as Promotion, SUM(SalesAmount) as 'Total Sales'
FROM FactResellerSales as fact INNER JOIN
             DimPromotion as promo ON fact.PromotionKey = promo.PromotionKey

WHERE promo.EnglishPromotionCategory = 'Reseller' AND
             (promo.EnglishPromotionType <> 'No Discount'
             AND promo.EnglishPromotionType NOT LIKE 'Volume Discount%') AND
             YEAR(fact.OrderDate) BETWEEN '2005' AND '2008'
GROUP BY promo.EnglishPromotionName
ORDER BY 2 DESC
```

NOTE: I looked at the FactResellerSales table OrderDate column and it looked like all dates were between 2005 and 2008, however, I left the specification in the code.

RESULTS

Promotion	Total Sales
Touring-1000 Promotion	581331.6288
Touring-3000 Promotion	443244.2043
Mountain-100 Clearance Sale	250927.6993
Road-650 Overstock	49986.0816
Mountain-500 Silver Clearance Sale	25899.1416
Sport Helmet Discount-2003	9100.9002
Sport Helmet Discount-2002	7448.8277

CONCLUSION

The Touring-1000 Promotion was the most successful in driving dollar sales between 2005 and 2008.

Problem 3

I am including two questions with the queries and results to answer them. This was because I felt I needed the practice and because I unintentionally had the advantage on the first problem. I used the correct assignment instructions for the rest of the problems.

Lisa Over
February 25, 2015
DW Assignment

QUESTION

What was the difference in total quantity of bicycle sales, by subcategory, from 2006 to 2007?

QUERY

```
USE tempdb;
IF OBJECT_ID('#Y1') IS NOT NULL
	DROP TABLE #Y1

IF OBJECT_ID('#Y2') IS NOT NULL
	DROP TABLE #Y2

IF OBJECT_ID('#Y3') IS NOT NULL
	DROP TABLE #Y3

USE [AdventureWorksDW2012]
GO

/*
Create two Temporary Tables #Y1 and #Y2 to hold the quantity of bikes sold in each
subcategory for the
specified years - one table for each year.

Create the #Y1 Temporary Table for 2006
*/
SELECT sum(fact.OrderQuantity) as 'Quantity', sub.EnglishProductSubcategoryName as
'Category'
INTO #Y1
FROM DimProductCategory as cat INNER JOIN
	DimProductSubcategory as sub ON
	cat.ProductCategoryKey = sub.ProductCategoryKey INNER JOIN
	DimProduct as prod ON sub.ProductSubcategoryKey = prod.ProductSubcategoryKey INNER
JOIN
	FactResellerSales as fact ON prod.ProductKey = fact.ProductKey INNER JOIN
	DimDate as dt ON fact.OrderDateKey = dt.DateKey
WHERE cat.EnglishProductCategoryName = 'Bikes' AND dt.CalendarYear in (2006)
GROUP BY sub.EnglishProductSubcategoryName

/*
Create the #Y2 Temporary Table for 2007
*/
SELECT sum(fact.OrderQuantity) as 'Quantity', sub.EnglishProductSubcategoryName as
'Category'
INTO #Y2
FROM DimProductCategory as cat INNER JOIN
	DimProductSubcategory as sub ON
	cat.ProductCategoryKey = sub.ProductCategoryKey INNER JOIN
	DimProduct as prod ON sub.ProductSubcategoryKey = prod.ProductSubcategoryKey INNER
JOIN
	FactResellerSales as fact ON prod.ProductKey = fact.ProductKey INNER JOIN
	DimDate as dt ON fact.OrderDateKey = dt.DateKey
WHERE cat.EnglishProductCategoryName = 'Bikes' AND dt.CalendarYear in (2007)
GROUP BY sub.EnglishProductSubcategoryName
```

Lisa Over
February 25, 2015
DW Assignment

```
/*
Combine the newly created temporary tables into another temporary table to
compare the difference in quantity of bikes sold by category and year.
 */
SELECT #Y1.Category as 'Bike Category', #Y1.Quantity as 'Quantity 2006',
       #Y2.Quantity as 'Quantity 2007', (#Y2.Quantity - #Y1.Quantity) as 'Qty Difference'
INTO #Y3
FROM #Y1 INNER JOIN #Y2 ON #Y1.Category = #Y2.Category

Select *
FROM #Y3
```

RESULTS

Bike Category	Quantity 2006	Quantity 2007	Qty Difference
Mountain Bikes	7260	9460	2200
Road Bikes	14971	14918	-53

QUESTION

For each reseller employee (those connected with reseller sales), what was the difference in sales quota and actual sales for the 4th quarter of 2007?

Note: I'm not sure what is meant by sales quota, but I am making the assumption that it is the expected amount of sales for employees.

QUERY

```
USE tempdb;
IF OBJECT_ID('#Sales') IS NOT NULL
       DROP TABLE #Sales

IF OBJECT_ID('#Emp') IS NOT NULL
       DROP TABLE #Emp

IF OBJECT_ID('#Quota') IS NOT NULL
       DROP TABLE #Quota

IF OBJECT_ID('#EmployeeSales') IS NOT NULL
       DROP TABLE #EmployeeSales

USE [AdventureWorksDW2012]
GO

/*
Create the #Sales Temporary Table for sales amount
*/
SELECT SUM(fact.SalesAmount) as 'SalesAmount', emp.EmployeeKey as 'EmployeeKey'
INTO #Sales
FROM DimEmployee as emp INNER JOIN
        FactResellerSales as fact ON emp.EmployeeKey = fact.EmployeeKey INNER JOIN
        DimDate as dt ON fact.OrderDateKey = dt.DateKey
WHERE dt.CalendarYear in (2007) AND dt.CalendarQuarter in (4)
GROUP BY emp.EmployeeKey
```

Lisa Over
February 25, 2015
DW Assignment

```
/*
Create the #Quota Temporary Table for sales quota
*/

SELECT SUM(fact.SalesAmountQuota) as 'SalesQuota', fact.EmployeeKey as 'EmployeeKey'
INTO #Quota
FROM DimEmployee as emp INNER JOIN
        FactSalesQuota as fact ON emp.EmployeeKey = fact.EmployeeKey INNER JOIN
        DimDate as dt ON fact.DateKey = dt.DateKey
WHERE dt.CalendarYear in (2007) AND dt.CalendarQuarter in (4)
GROUP BY fact.EmployeeKey

/*
Create the #Emp Temporary Table for employee names
*/
SELECT emp.FirstName as 'FirstName', emp.MiddleName as 'MiddleName', emp.LastName as
       'LastName', emp.EmployeeKey as 'EmployeeKey'
INTO #Emp
FROM DimEmployee as emp


/*
Combine the newly created temporary tables into another temporary table to
show sales amount, sales quota, and the difference per employee in the 4th quarter of
2007.
 */
SELECT TOP 99.99 PERCENT WITH TIES (ISNULL(#Emp.FirstName + ' ', '') +
ISNULL(#Emp.MiddleName + ' ', '') + COALESCE(#Emp.LastName,'')) as 'Employee Name',
       #Sales.SalesAmount as 'Sales Amount', #Quota.SalesQuota as 'Sales Quota',
       (#Quota.SalesQuota - #Sales.SalesAmount) as 'Quota to Sales Difference'
INTO #EmployeeSales
FROM #Sales JOIN #Emp ON #Sales.EmployeeKey = #Emp.EmployeeKey JOIN #Quota ON
       #Sales.EmployeeKey = #Quota.EmployeeKey
ORDER BY 'Quota to Sales Difference'

Select *
FROM #EmployeeSales
```

RESULTS

Employee Name	Sales Amount	Sales Quota	Quota to Sales Difference
David R Campbell	368726.001	372000.00	3273.999
Syed E Abbas	34019.172	40000.00	5980.828
Stephen Y Jiang	92752.343	116000.00	23247.657
Lynn N Tsoflias	362527.3866	389000.00	26472.6134
Tete A Mensa-Annan	346625.926	380000.00	33374.074
Garrett R Vargas	419307.7273	453000.00	33692.2727
Shu K Ito	635929.6466	684000.00	48070.3534
Amy E Alberts	275681.9791	324000.00	48318.0209
Rachel B Valdez	473444.2881	566000.00	92555.7119
Tsvi Michael Reiter	570076.0237	675000.00	104923.9763

Lisa Over
February 25, 2015
DW Assignment

Problem 4

Question 1

Table	Data Space
DimCustomer	12.258 MB
DimGeography	0.195 MB
DimSalesTerritory	1.727 MB
TOTAL	**14.18**

Question 2

Code submitted as specified by uploading the *Problem4.sql* file.

Question 3

Table	Data Space
DimCustomer2	12.109 MB

The required data space from converting the three 3NF tables DimCustomer, DimGeography, and DimSalesTerritory to the 1NF table DimCustomer2 was decreased by 14.61%.

Problem 5

Hadoop

Hadoop, an open-source software under the Apache Software Foundation, consists of several modules that allow users to facilitate a distributed file system of large data sets. In a distributed file system, files are stored on many computers but are accessed as a whole, seamless system. The software modules are designed to handle hardware failures automatically because of how often failures occur. The Hadoop software addresses the problem of handling the growing number and size of data sets by providing a framework for storing and managing them. The Hadoop modules YARN and MapReduce provide users with a means to schedule jobs, manage clusters of machines, and perform parallel processing of the large data sets across machines. Hadoop was originally developed by Doug Cutting and Mike Cafarella in 2005 to support the distribution of the Nutch search engine project.

http://hadoop.apache.org/#What+Is+Apache+Hadoop%3F
http://en.wikipedia.org/wiki/Apache_Hadoop

Hive

Hive is an open-source data warehouse software that was initially developed by Facebook and then started as a subproject of Hadoop under the Apache Software Foundation. It provides extended support for querying and managing large datasets that are distributed across many computers. It graduated to be its own project, however, it still runs on top of Hadoop to provide data summarization, query, and analysis. Hive provides its own query language, HiveQL, for querying the data and transparently converts

queries to map/reduce. It is also flexible in that it allows programmers to use traditional mappers and reducers when it is inconvenient or inefficient to use HiveQL.

http://hive.apache.org
http://en.wikipedia.org/wiki/Apache_Hive

Pig and Piglatin

Pig is a high-level platform developed by Yahoo Research in 2006 for their researchers to create makeshift map/reduce programs. Pig was moved to be a project under the Apache Software Foundation in 2007. Pig has its own language, Piglatin, and it is this language that makes it high-level. Similar to SQL for RDBMS, Piglatin takes the programming from the Java MapReduce idiom and abstracts it into a high-level language. Piglatin supports User Defined Functions (UDF), which users can write using Java, Python, JavaScript, Ruby, or Groovy. Piglatin language basics – conventions, reserved words, operators, etc. – are explained at https://pig.apache.org/docs/r0.11.1/basic.html.

http://pig.apache.org
http://en.wikipedia.org/wiki/Pig_(programming_tool)

Zookeeper

Zookeeper is an open-source distributed configuration service, synchronization service, and naming registry for large distributed systems with an interface for accessing the services. Once a sub-project of Hadoop, Zookeeper is now its own project under the Apache Software Foundation. The Zookeeper project supports high availability by providing redundant services, i.e., if one Zookeeper fails to respond to a client, the client may ask a different Zookeeper master. The configuration management service establishes and maintains consistency of the distributed system's performance, design, functional and physical requirements, and operation. The synchronization service serves to coordinate events of the system that operate in unison. The naming registry stores, organizes, and provides access to information in the distributed system.

http://zookeeper.apache.org
http://zookeeper.apache.org/doc/trunk/
http://en.wikipedia.org/wiki/Apache_ZooKeeper
http://en.wikipedia.org/wiki/Configuration_management
http://en.wikipedia.org/wiki/Synchronization
http://en.wikipedia.org/wiki/Directory_service

Spotify

Spotify, a digital music service, uses Hadoop for content generation, data aggregation, reporting and analysis. They started with a 37-node cluster about 5 years ago but now have a 690-node cluster to store data on millions of daily users. They claim to be a data-driven company that uses their data for every thing including

- Reporting to labels, licensures, partners, and advertisers.

- Analyzing business growth, user behavior, key performance indicators, measures taken at various signup events to track how customer move through the process, testing website optimization changes before going live, etc.
- Analyzing operations, which involves identifying root causes of faults or problems; reducing the time it takes to collect and store data, to analyze the data, and to react and take action based on the data; improving capacity through the planning of servers, people, and bandwidth.

Spotify wrote and open-sourced their own scheduler, Luigi, because MapReduce was too limited for what they wanted, which was to chain jobs. They say that Hadoop "brought them very far;" a cheap RDBMS would not have been able to handle their current volume of data. However, they also say that Hadoop is not a "silver bullet;" it is a system that needs "love and care."

http://wiki.apache.org/hadoop/PoweredBy#D
http://files.meetup.com/5139282/SHUG%201%20-%20Hadoop%20at%20Spotify.pdf

---

Forward3D

Forward3D is a search and display marketing firm that runs cross-market and multilingual campaigns from offices in London, New York, Shanghai, and Seoul. Forward3D attributes its significant growth over the past 10 years to its best practice implementation, **approach to technology**, and core values. They would not be able to handle the amount of data they need to handle without Hadoop. Their technology approach includes the use of Hadoop to run almost all of their daily processing, which includes daily batch ETL, log analysis, data mining, and machine learning. They also use Hive for further ad-hoc analysis. The majority of their processing is automated with Ruby and their own open-source software, Mandy. Mandy makes it easier to run MapReduce by providing a simple structure for writing code. They automate the analysis, storage, and processing of approximately 80 GB of data daily from 9 million clicks across their campaigns. Their storage and processing utilizes a 5-machine cluster with 8 cores and 5 TB of storage per machine. They also have 19 virtual machines with 2 cores per machine and 30 TB of storage.

https://wiki.apache.org/hadoop/PoweredBy
http://oobaloo.co.uk/articles/2010/1/12/mapreduce-with-hadoop-and-ruby.html
http://www.forward3d.com