SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Jordan Shields>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

ALTER TRIGGER MAXBRANDQTY

ON PRODUCT

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE QTYCHECKER CURSOR FOR

SELECT BRAND\_ID, AVG(PROD\_QOH)

FROM INSERTED

GROUP BY BRAND\_ID

DECLARE @QOH DECIMAL(10,0)

DECLARE @BRAND\_ID INT

IF(EXISTS (SELECT \* FROM INSERTED))

BEGIN

OPEN QTYCHECKER

FETCH NEXT FROM QTYCHECKER INTO @BRAND\_ID, @QOH

WHILE (@@FETCH\_STATUS = 0)

BEGIN

IF (@QOH > (SELECT AVG(PROD\_QOH)

FROM PRODUCT))

BEGIN

PRINT ('BRAND ID "' + CAST(@BRAND\_ID AS VARCHAR(15)) + '"CONTAINS MORE QOH THAN AVG QOH')

END

ELSE

BEGIN

SET @QOH = (SELECT AVG(PROD\_QOH)

FROM PRODUCT) - @QOH;

PRINT ('PURCHASE ' + (CAST(@QOH AS VARCHAR(20))) + ' MORE ITEMS TO MEET AVG BRAND QOH')

END

FETCH NEXT FROM QTYCHECKER INTO @BRAND\_ID, @QOH

END

CLOSE QTYCHECKER

DEALLOCATE QTYCHECKER

END

END

GO

USE [CIS31031]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[A10] Script Date: 4/16/2014 9:27:24 AM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: JORDAN SHIELDS

-- Create date: 4/21/2014

-- Description: CIS 310-01

-- ASSIGNMENT 10

-- =============================================

CREATE PROCEDURE [dbo].[A10]

-- Add the parameters for the stored procedure here

@LAST\_NAME NVARCHAR(20) = NULL

AS

BEGIN

DECLARE @CUS\_CODE INT

DECLARE @COMPARE\_CODE INT

DECLARE @BRAN\_NAME NVARCHAR(100)

DECLARE @CUS\_LNAME NVARCHAR(20)

DECLARE CUSTOMERS CURSOR FOR

SELECT C.CUST\_CODE, C.CUST\_LNAME, B.BRAND\_NAME

FROM LGBRAND B

INNER JOIN LGPRODUCT P ON P.BRAND\_ID = B.BRAND\_ID

INNER JOIN LGLINE L ON L.PROD\_SKU = P.PROD\_SKU

INNER JOIN LGINVOICE I ON I.INV\_NUM = L.INV\_NUM

INNER JOIN LGCUSTOMER C ON C.CUST\_CODE = I.CUST\_CODE

GROUP BY C.CUST\_CODE, C.CUST\_LNAME, B.BRAND\_NAME

ORDER BY C.CUST\_LNAME, C.CUST\_CODE, B.BRAND\_NAME

IF(@LAST\_NAME IS NULL)

BEGIN

OPEN CUSTOMERS

FETCH NEXT FROM CUSTOMERS INTO @CUS\_CODE, @CUS\_LNAME, @BRAN\_NAME

WHILE (@@FETCH\_STATUS = 0)

BEGIN

PRINT @CUS\_LNAME

SET @COMPARE\_CODE = @CUS\_CODE

WHILE (@@FETCH\_STATUS = 0)

BEGIN

PRINT (' ' + @BRAN\_NAME)

FETCH NEXT FROM CUSTOMERS INTO @CUS\_CODE, @CUS\_LNAME, @BRAN\_NAME

IF (@CUS\_CODE <> @COMPARE\_CODE)

BREAK

ELSE

CONTINUE

END

END

CLOSE CUSTOMERS

END

ELSE IF (@LAST\_NAME IN (SELECT C.CUST\_LNAME

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON I.CUST\_CODE = C.CUST\_CODE

GROUP BY C.CUST\_CODE, C.CUST\_LNAME))

BEGIN

OPEN CUSTOMERS

FETCH NEXT FROM CUSTOMERS INTO @CUS\_CODE, @CUS\_LNAME, @BRAN\_NAME

WHILE (@@FETCH\_STATUS = 0)

BEGIN

IF (@LAST\_NAME = @CUS\_LNAME)

BEGIN

PRINT @CUS\_LNAME

SET @COMPARE\_CODE = @CUS\_CODE

WHILE (@@FETCH\_STATUS = 0)

BEGIN

PRINT (' ' + @BRAN\_NAME)

FETCH NEXT FROM CUSTOMERS INTO @CUS\_CODE, @CUS\_LNAME, @BRAN\_NAME

IF (@CUS\_CODE <> @COMPARE\_CODE)

BREAK

ELSE

CONTINUE

END

END

ELSE IF (@@FETCH\_STATUS = 0)

FETCH NEXT FROM CUSTOMERS INTO @CUS\_CODE, @CUS\_LNAME, @BRAN\_NAME

END

CLOSE CUSTOMERS

END

ELSE IF (@LAST\_NAME IN (SELECT CUST\_LNAME FROM LGCUSTOMER))

BEGIN

PRINT ('NO CUSTOMER WITH LAST NAME "' + @LAST\_NAME + '" HAS MADE A PURCHASE')

END

ELSE

BEGIN

PRINT ('CUSTOMER LAST NAME "' + @LAST\_NAME + '" DOES NOT EXIST IN CURRENT DATABASE')

END

DEALLOCATE CUSTOMERS

END

--TEST CODE

--EXEC A10

--EXEC A10 ABEL

--EXEC A10 HAYNES

--EXEC A10 BL

-- Jordan Shields

-- CIS 310-01

-- 04-14-14

-- PART 1: Write SQL code to update INV\_TOTAL and CUST\_BALANCE so that those values correctly reflect the values in LGLINE.

UPDATE I

SET I.INV\_TOTAL = (SELECT SUM(L.LINE\_QTY \* L.LINE\_PRICE)

FROM LGLINE L

WHERE L.INV\_NUM = I.INV\_NUM

GROUP BY INV\_NUM)

FROM LGINVOICE I

UPDATE C

SET C.CUST\_BALANCE = (SELECT SUM(I.INV\_TOTAL)

FROM LGINVOICE I

WHERE I.CUST\_CODE = C.CUST\_CODE

GROUP BY CUST\_CODE)

FROM LGCUSTOMER C

-- PART 2 -- TEST

SELECT I.INV\_NUM , I.INV\_TOTAL, C.CUST\_CODE, C.CUST\_BALANCE

FROM LGINVOICE I

INNER JOIN LGCUSTOMER C ON C.CUST\_CODE = I.CUST\_CODE

WHERE I.INV\_NUM = 104

SELECT INV\_NUM, LINE\_NUM , PROD\_SKU, LINE\_QTY, LINE\_PRICE

FROM LGLINE

WHERE INV\_NUM = 104

INSERT INTO LGLINE (INV\_NUM, LINE\_NUM, PROD\_SKU, LINE\_QTY, LINE\_PRICE)

VALUES(104, 4, '6041-PBS', 16, 15.00)

DELETE FROM LGLINE

WHERE INV\_NUM = 104 AND LINE\_NUM = 4

SELECT L.INV\_NUM, SUM(L.LINE\_QTY \* L.LINE\_PRICE), I.INV\_TOTAL

FROM LGLINE L

INNER JOIN LGINVOICE I ON I.INV\_NUM = L.INV\_NUM

WHERE L.INV\_NUM = I.INV\_NUM AND I.INV\_NUM = 104

GROUP BY L.INV\_NUM, I.INV\_TOTAL

SELECT C.CUST\_CODE, SUM(I.INV\_TOTAL), C.CUST\_BALANCE

FROM LGINVOICE I

INNER JOIN LGCUSTOMER C ON C.CUST\_CODE = I.CUST\_CODE

WHERE I.CUST\_CODE = C.CUST\_CODE AND C.CUST\_CODE = 88

GROUP BY C.CUST\_CODE, C.CUST\_BALANCE

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: JORDAN SHIELDS

-- Create date: 4/14/2014

-- Description: CIS 310-01

-- =============================================

CREATE TRIGGER LINE\_UPDATE

ON LGLINE

AFTER INSERT,DELETE,UPDATE

AS

BEGIN

DECLARE @INV\_NUM INT

DECLARE @INV\_TOTAL DECIMAL(12,2)

DECLARE INSERT\_INVOICE CURSOR FOR

SELECT INV\_NUM, SUM(LINE\_QTY \* LINE\_PRICE)

FROM INSERTED

GROUP BY INV\_NUM

DECLARE DELETED\_INVOICE CURSOR FOR

SELECT INV\_NUM, SUM(LINE\_QTY \* LINE\_PRICE)

FROM DELETED

GROUP BY INV\_NUM

IF(EXISTS (SELECT \* FROM INSERTED))

BEGIN

OPEN INSERT\_INVOICE

FETCH NEXT FROM INSERT\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

WHILE (@@FETCH\_STATUS = 0)

BEGIN

UPDATE LGINVOICE

SET INV\_TOTAL = INV\_TOTAL + @INV\_TOTAL

WHERE INV\_NUM = @INV\_NUM

UPDATE C

SET C.CUST\_BALANCE = C.CUST\_BALANCE + @INV\_TOTAL

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON I.CUST\_CODE = C.CUST\_CODE

WHERE I.INV\_NUM = @INV\_NUM AND

I.CUST\_CODE = C.CUST\_CODE

FETCH NEXT FROM INSERT\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

END

CLOSE INSERT\_INVOICE

DEALLOCATE INSERT\_INVOICE

END

IF(EXISTS (SELECT \* FROM DELETED))

BEGIN

OPEN DELETED\_INVOICE

FETCH NEXT FROM DELETED\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

WHILE (@@FETCH\_STATUS = 0)

BEGIN

UPDATE LGINVOICE

SET INV\_TOTAL = INV\_TOTAL - @INV\_TOTAL

WHERE INV\_NUM = @INV\_NUM

UPDATE C

SET C.CUST\_BALANCE = C.CUST\_BALANCE - @INV\_TOTAL

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON I.CUST\_CODE = C.CUST\_CODE

WHERE I.INV\_NUM = @INV\_NUM AND

I.CUST\_CODE = C.CUST\_CODE

FETCH NEXT FROM DELETED\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

END

CLOSE DELETED\_INVOICE

DEALLOCATE DELETED\_INVOICE

END

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: JORDAN SHIELDS

-- Create date: 4/14/2014

-- Description: CIS 310-01

-- =============================================

CREATE TRIGGER LINE\_UPDATE

ON LGLINE

AFTER INSERT,DELETE,UPDATE

AS

BEGIN

DECLARE @INV\_NUM INT

DECLARE @INV\_TOTAL DECIMAL(12,2)

DECLARE INSERT\_INVOICE CURSOR FOR

SELECT INV\_NUM, SUM(LINE\_QTY \* LINE\_PRICE)

FROM INSERTED

GROUP BY INV\_NUM

DECLARE DELETED\_INVOICE CURSOR FOR

SELECT INV\_NUM, SUM(LINE\_QTY \* LINE\_PRICE)

FROM DELETED

GROUP BY INV\_NUM

IF(EXISTS (SELECT \* FROM INSERTED))

BEGIN

OPEN INSERT\_INVOICE

FETCH NEXT FROM INSERT\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

WHILE (@@FETCH\_STATUS = 0)

BEGIN

UPDATE LGINVOICE

SET INV\_TOTAL = INV\_TOTAL + @INV\_TOTAL

WHERE INV\_NUM = @INV\_NUM

UPDATE C

SET C.CUST\_BALANCE = C.CUST\_BALANCE + @INV\_TOTAL

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON I.CUST\_CODE = C.CUST\_CODE

WHERE I.INV\_NUM = @INV\_NUM AND

I.CUST\_CODE = C.CUST\_CODE

FETCH NEXT FROM INSERT\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

END

CLOSE INSERT\_INVOICE

DEALLOCATE INSERT\_INVOICE

END

IF(EXISTS (SELECT \* FROM DELETED))

BEGIN

OPEN DELETED\_INVOICE

FETCH NEXT FROM DELETED\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

WHILE (@@FETCH\_STATUS = 0)

BEGIN

UPDATE LGINVOICE

SET INV\_TOTAL = INV\_TOTAL - @INV\_TOTAL

WHERE INV\_NUM = @INV\_NUM

UPDATE C

SET C.CUST\_BALANCE = C.CUST\_BALANCE - @INV\_TOTAL

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON I.CUST\_CODE = C.CUST\_CODE

WHERE I.INV\_NUM = @INV\_NUM AND

I.CUST\_CODE = C.CUST\_CODE

FETCH NEXT FROM DELETED\_INVOICE INTO @INV\_NUM, @INV\_TOTAL

END

CLOSE DELETED\_INVOICE

DEALLOCATE DELETED\_INVOICE

END

END

GO

CREATE TABLE EMPLOYEE

(

EMP\_NUM INT NOT NULL,

EMP\_FNAME NVARCHAR(20),

EMP\_LNAME NVARCHAR(25),

EMP\_EMAIL NVARCHAR(25),

EMP\_PHONE NVARCHAR(20),

EMP\_HIREDATE DATETIME,

EMP\_TITLE NVARCHAR(45),

EMP\_COMM DECIMAL(2,2),

DEPT\_NUM INT

)

ALTER TABLE EMPLOYEE

ADD CONSTRAINT PK\_EMPLOYEE PRIMARY KEY (EMP\_NUM)

CREATE TABLE BRAND

(

BRAND\_ID INT NOT NULL,

BRAND\_NAME NVARCHAR(100),

BRAND\_TYPE NVARCHAR(20)

)

ALTER TABLE BRAND

ADD CONSTRAINT PK\_BRAND PRIMARY KEY (BRAND\_ID)

CREATE TABLE CUSTOMER

(

CUST\_CODE INT NOT NULL,

CUST\_FNAME NVARCHAR(20),

CUST\_LNAME NVARCHAR(20),

CUST\_STREET NVARCHAR(70),

CUST\_CITY NVARCHAR(50),

CUST\_STATE NVARCHAR(2),

CUST\_ZIP NVARCHAR(5),

CUST\_BALANCE DECIMAL(8,2)

)

ALTER TABLE CUSTOMER

ADD CONSTRAINT PK\_CUSTOMER PRIMARY KEY (CUST\_CODE)

CREATE TABLE DEPARTMENT

(

DEPT\_NUM INT NOT NULL,

DEPT\_NAME NVARCHAR(50),

DEPT\_MAIL\_BOX NVARCHAR(3),

DEPT\_PHONE NVARCHAR(9),

EMP\_NUM INT

)

ALTER TABLE DEPARTMENT

ADD CONSTRAINT PK\_DEPARTMENT PRIMARY KEY (DEPT\_NUM),

CONSTRAINT FK\_DEPARTMENT\_EMPLOYEE FOREIGN KEY (EMP\_NUM) REFERENCES EMPLOYEE

ALTER TABLE EMPLOYEE

ADD CONSTRAINT FK\_EMPLOYEE\_DEPARTMENT FOREIGN KEY (DEPT\_NUM) REFERENCES DEPARTMENT

CREATE TABLE INVOICE

(

INV\_NUM INT NOT NULL,

INV\_DATE DATETIME,

CUST\_CODE INT,

INV\_TOTAL DECIMAL(11,2),

EMPLOYEE\_ID INT

)

ALTER TABLE INVOICE

ADD CONSTRAINT PK\_INVOICE PRIMARY KEY (INV\_NUM),

CONSTRAINT FK\_INVOICE\_EMPLOYEE FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE (EMP\_NUM),

CONSTRAINT FK\_INVOICE\_CUSTOMER FOREIGN KEY (CUST\_CODE) REFERENCES CUSTOMER

CREATE TABLE LINE

(

INV\_NUM INT NOT NULL,

LINE\_NUM INT NOT NULL,

PROD\_SKU NVARCHAR(15),

LINE\_QTY FLOAT,

LINE\_PRICE DECIMAL(8,2)

)

ALTER TABLE LINE

ADD CONSTRAINT PK\_LINE PRIMARY KEY (INV\_NUM, LINE\_NUM),

CONSTRAINT FK\_LINE\_INVOICE FOREIGN KEY (INV\_NUM) REFERENCES INVOICE

CREATE TABLE PRODUCT

(

PROD\_SKU NVARCHAR(15) NOT NULL,

PROD\_DESCRIPT NVARCHAR(255),

PROD\_TYPE NVARCHAR(255),

PROD\_BASE NVARCHAR(255),

PROD\_CATEGORY NVARCHAR(255),

PROD\_PRICE DECIMAL(10,2),

PROD\_QOH DECIMAL(10,0),

PROD\_MIN DECIMAL(10,0),

BRAND\_ID INT

)

ALTER TABLE PRODUCT

ADD CONSTRAINT PK\_PRODUCT PRIMARY KEY (PROD\_SKU),

CONSTRAINT FK\_PRODUCT\_BRAND FOREIGN KEY (BRAND\_ID) REFERENCES BRAND

ALTER TABLE LINE

ADD CONSTRAINT FK\_LINE\_PRODUCT FOREIGN KEY (PROD\_SKU) REFERENCES PRODUCT

CREATE TABLE SALARY\_HISTORY

(

EMP\_NUM INT NOT NULL,

SAL\_FROM DATETIME NOT NULL,

SAL\_END DATETIME,

SAL\_AMOUNT DECIMAL(10,2)

)

ALTER TABLE SALARY\_HISTORY

ADD CONSTRAINT PK\_SALARY\_HISTORY PRIMARY KEY (EMP\_NUM, SAL\_FROM),

CONSTRAINT FK\_SALARY\_HISTORY\_EMPLOYEE FOREIGN KEY (EMP\_NUM) REFERENCES EMPLOYEE

CREATE TABLE VENDOR

(

VEND\_ID INT NOT NULL,

VEND\_NAME NVARCHAR(255),

VEND\_STREET NVARCHAR(50),

VEND\_CITY NVARCHAR(50),

VEND\_STATE NVARCHAR(2),

VEND\_ZIP NVARCHAR(5)

)

ALTER TABLE VENDOR

ADD CONSTRAINT PK\_VENDOR PRIMARY KEY (VEND\_ID)

CREATE TABLE SUPPLIES

(

PROD\_SKU NVARCHAR(15) NOT NULL,

VEND\_ID INT NOT NULL

)

ALTER TABLE SUPPLIES

ADD CONSTRAINT PK\_SUPPLIES PRIMARY KEY (PROD\_SKU, VEND\_ID),

CONSTRAINT FK\_SUPPLIES\_PRODUCT FOREIGN KEY (PROD\_SKU) REFERENCES PRODUCT,

CONSTRAINT FK\_SUPPLIES\_VENDOR FOREIGN KEY (VEND\_ID) REFERENCES VENDOR

-- Part 2

INSERT INTO EMPLOYEE (EMP\_NUM, EMP\_FNAME, EMP\_LNAME, EMP\_EMAIL, EMP\_PHONE, EMP\_HIREDATE, EMP\_TITLE, EMP\_COMM)

SELECT EMP\_NUM, EMP\_FNAME, EMP\_LNAME, EMP\_EMAIL, EMP\_PHONE, EMP\_HIREDATE, EMP\_TITLE, EMP\_COMM

FROM LGEMPLOYEE

INSERT INTO BRAND

SELECT \*

FROM LGBRAND

INSERT INTO CUSTOMER

SELECT \*

FROM LGCUSTOMER

INSERT INTO VENDOR

SELECT \*

FROM LGVENDOR

INSERT INTO DEPARTMENT

SELECT \*

FROM LGDEPARTMENT

UPDATE E

SET E.DEPT\_NUM = L.DEPT\_NUM

FROM EMPLOYEE E

INNER JOIN LGEMPLOYEE L ON E.EMP\_NUM = L.EMP\_NUM

WHERE E.EMP\_NUM = L.EMP\_NUM

INSERT INTO INVOICE

SELECT \*

FROM LGINVOICE

INSERT INTO PRODUCT

SELECT \*

FROM LGPRODUCT

INSERT INTO LINE

SELECT \*

FROM LGLINE

INSERT INTO SALARY\_HISTORY

SELECT \*

FROM LGSALARY\_HISTORY

INSERT INTO SUPPLIES

SELECT \*

FROM LGSUPPLIES

--Q44

SELECT DEPT\_NAME

FROM LGDEPARTMENT

--Q45

SELECT PROD\_SKU, PROD\_DESCRIPT, PROD\_TYPE, PROD\_BASE, PROD\_CATEGORY, PROD\_PRICE

FROM LGPRODUCT

WHERE PROD\_BASE = 'WATER' AND

PROD\_CATEGORY = 'SEALER'

--Q46

SELECT [EMP\_FNAME], [EMP\_LNAME], [EMP\_EMAIL]

FROM [dbo].[LGEMPLOYEE]

WHERE CAST([EMP\_HIREDATE] AS DATE) BETWEEN

'2001-01-01' AND '2010-12-31'

ORDER BY [EMP\_LNAME], [EMP\_FNAME]

--Q47

SELECT [EMP\_FNAME], [EMP\_LNAME], [EMP\_PHONE], [EMP\_TITLE], [DEPT\_NUM]

FROM [dbo].[LGEMPLOYEE]

WHERE [DEPT\_NUM] = 300 OR

[EMP\_TITLE] = 'CLERK I'

ORDER BY [EMP\_LNAME], [EMP\_FNAME]

--Q48

SELECT [dbo].[LGSALARY\_HISTORY].\*, EMP\_LNAME, EMP\_FNAME

FROM [dbo].[LGSALARY\_HISTORY]

INNER JOIN [dbo].[LGEMPLOYEE] ON [dbo].[LGSALARY\_HISTORY].[EMP\_NUM] = [dbo].[LGEMPLOYEE].EMP\_NUM

WHERE [dbo].[LGSALARY\_HISTORY].EMP\_NUM = 83731

OR [dbo].[LGSALARY\_HISTORY].EMP\_NUM = 83745

OR [dbo].[LGSALARY\_HISTORY].EMP\_NUM = 84039

ORDER BY EMP\_NUM, SAL\_FROM

--Q49

SELECT DISTINCT [CUST\_FNAME], [CUST\_LNAME], [CUST\_STREET], [CUST\_CITY], [CUST\_STATE], [CUST\_ZIP]

FROM [dbo].[LGCUSTOMER] C

INNER JOIN [dbo].[LGINVOICE] I ON C.[CUST\_CODE] = I.CUST\_CODE

INNER JOIN LGLINE L ON I.INV\_NUM = L.INV\_NUM

INNER JOIN LGPRODUCT P ON L.PROD\_SKU = P.PROD\_SKU

INNER JOIN LGBRAND B ON P.BRAND\_ID = B.BRAND\_ID

WHERE B.BRAND\_NAME = 'FORESTERS BEST' AND

P.PROD\_CATEGORY = 'TOP COAT' AND

CAST(I.INV\_DATE AS DATE) BETWEEN '2011-07-15' AND '2011-07-31'

ORDER BY C.CUST\_STATE, C.CUST\_LNAME, C.CUST\_FNAME

--Q50

SELECT E.[EMP\_NUM], E.EMP\_LNAME, E.EMP\_EMAIL, E.EMP\_TITLE, D.DEPT\_NAME

FROM LGEMPLOYEE E

LEFT JOIN LGDEPARTMENT D ON E.DEPT\_NUM = D.DEPT\_NUM

WHERE E.EMP\_TITLE LIKE '%ASSOCIATE'

ORDER BY D.DEPT\_NAME, E.EMP\_TITLE

--Q51

SELECT B.BRAND\_NAME, COUNT(P.BRAND\_ID) TOTAL\_COUNT

FROM LGBRAND B

INNER JOIN LGPRODUCT P ON B.BRAND\_ID = P.BRAND\_ID

GROUP BY B.BRAND\_NAME

--Q52

SELECT PROD\_CATEGORY, COUNT(PROD\_BASE) TOTAL\_COUNT

FROM LGPRODUCT

WHERE PROD\_BASE = 'WATER'

GROUP BY PROD\_CATEGORY

--Q53

SELECT PROD\_BASE, PROD\_TYPE, COUNT(PROD\_SKU) TOTAL\_PRODUCTS

FROM LGPRODUCT

GROUP BY PROD\_BASE, PROD\_TYPE

ORDER BY PROD\_BASE

--Q54

SELECT BRAND\_ID, SUM(PROD\_QOH) TOTAL\_INV

FROM LGPRODUCT

GROUP BY BRAND\_ID

ORDER BY BRAND\_ID DESC

--Q55

SELECT B.BRAND\_ID, B.BRAND\_NAME, CAST(AVG(P.PROD\_PRICE)AS DECIMAL(30,2)) AVG\_PRICE

FROM LGBRAND B

INNER JOIN LGPRODUCT P ON B.BRAND\_ID = P.BRAND\_ID

GROUP BY B.BRAND\_ID, B.BRAND\_NAME

ORDER BY B.BRAND\_NAME

--Q56

SELECT DEPT\_NUM, MAX(CAST(EMP\_HIREDATE AS DATE)) MOST\_RECENT\_HIRE

FROM LGEMPLOYEE

GROUP BY DEPT\_NUM

ORDER BY DEPT\_NUM

--Q57

SELECT E.EMP\_NUM, E.EMP\_FNAME, E.EMP\_LNAME, MAX(S.SAL\_AMOUNT) LARGEST\_SALARY

FROM LGEMPLOYEE E

INNER JOIN LGSALARY\_HISTORY S ON E.EMP\_NUM = S.EMP\_NUM

WHERE E.DEPT\_NUM = 200

GROUP BY E.EMP\_NUM, E.EMP\_FNAME, E.EMP\_LNAME

ORDER BY MAX(S.SAL\_AMOUNT) DESC

--Q58

SELECT C.CUST\_CODE, C.CUST\_FNAME, C.CUST\_LNAME, SUM(I.INV\_TOTAL) INVOICE\_SUM

FROM LGCUSTOMER C

INNER JOIN LGINVOICE I ON C.CUST\_CODE = I.CUST\_CODE

GROUP BY C.CUST\_CODE, C.CUST\_FNAME, C.CUST\_LNAME

HAVING SUM(I.INV\_TOTAL) > 1500

ORDER BY SUM(INV\_TOTAL) DESC

--Q59

SELECT D.DEPT\_NUM, D.DEPT\_NAME, D.DEPT\_PHONE, E.EMP\_NUM, E.EMP\_LNAME

FROM LGDEPARTMENT D

INNER JOIN LGEMPLOYEE E ON D.EMP\_NUM = E.EMP\_NUM

ORDER BY D.DEPT\_NAME

--Q60

SELECT V.VEND\_ID, V.VEND\_NAME, B.BRAND\_NAME, COUNT(S.PROD\_SKU) NUM\_PRODUCTS

FROM LGVENDOR V

INNER JOIN LGSUPPLIES S ON V.VEND\_ID = S.VEND\_ID

INNER JOIN LGPRODUCT P ON S.PROD\_SKU = P.PROD\_SKU

INNER JOIN LGBRAND B ON P.BRAND\_ID = B.BRAND\_ID

GROUP BY V.VEND\_ID, V.VEND\_NAME, B.BRAND\_NAME

ORDER BY V.VEND\_NAME, B.BRAND\_NAME

--Q61

SELECT E.EMP\_NUM, E.EMP\_LNAME, E.EMP\_FNAME, SUM(I.INV\_TOTAL) TOTAL\_INVOICES

FROM LGEMPLOYEE E

INNER JOIN LGINVOICE I ON E.EMP\_NUM = I.EMPLOYEE\_ID

GROUP BY E.EMP\_NUM, E.EMP\_LNAME, E.EMP\_FNAME

ORDER BY E.EMP\_LNAME, E.EMP\_FNAME

--Q62

SELECT MAX(AVG\_PRICE) MAX\_AVG

FROM (SELECT CAST(AVG(PROD\_PRICE) AS DECIMAL (30,2)) AVG\_PRICE

FROM LGPRODUCT

GROUP BY BRAND\_ID) Q1

--Q63

SELECT TOP 1 P.BRAND\_ID, B.BRAND\_NAME, B.BRAND\_TYPE, CAST(AVG(P.PROD\_PRICE) AS DECIMAL (30,2)) AVG\_PRICE

FROM LGPRODUCT P

INNER JOIN LGBRAND B ON P.BRAND\_ID = B.BRAND\_ID

GROUP BY P.BRAND\_ID, B.BRAND\_NAME, B.BRAND\_TYPE

ORDER BY AVG\_PRICE DESC

--Q63-2

SELECT P.BRAND\_ID, B.BRAND\_NAME, B.BRAND\_TYPE, CAST(AVG(P.PROD\_PRICE) AS DECIMAL (30,2)) AVG\_PRICE

FROM LGPRODUCT P

INNER JOIN LGBRAND B ON P.BRAND\_ID = B.BRAND\_ID

GROUP BY P.BRAND\_ID, B.BRAND\_NAME, B.BRAND\_TYPE

HAVING CAST(AVG(P.PROD\_PRICE) AS DECIMAL (30,2)) = (SELECT TOP 1 CAST(AVG(P.PROD\_PRICE) AS DECIMAL (30,2)) AVG\_PRICE

FROM LGPRODUCT P

GROUP BY P.BRAND\_ID

ORDER BY AVG\_PRICE DESC)

--Q64

SELECT (E.EMP\_FNAME + ' ' + E.EMP\_LNAME) DEPARTMENT\_MANAGER, D.DEPT\_NAME, D.DEPT\_PHONE, (E2.EMP\_FNAME + ' ' + E2.EMP\_LNAME) EMPLOYEE,

(C.CUST\_FNAME + ' ' + C.CUST\_LNAME) CUSTOMER, CAST(I.INV\_DATE AS DATE) INV\_DATE, I.INV\_TOTAL

FROM LGDEPARTMENT D

INNER JOIN LGEMPLOYEE E ON D.EMP\_NUM = E.EMP\_NUM

INNER JOIN LGEMPLOYEE E2 ON D.DEPT\_NUM = E2.DEPT\_NUM

INNER JOIN LGINVOICE I ON I.EMPLOYEE\_ID = E2.EMP\_NUM

INNER JOIN LGCUSTOMER C ON C.CUST\_CODE = I.CUST\_CODE

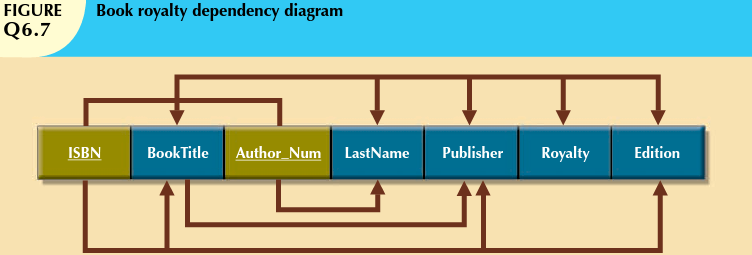
WHERE C.CUST\_LNAME = 'HAGAN'

AND CAST(I.INV\_DATE AS DATE) = '2011-05-18'

Jordan Shields

CIS 310-01

03/03/2014



**Part one: List all the undesirable dependencies and explain why they are undesirable.**

**Partial dependencies:**

* ISBN 🡪 (BookTitle, Publisher, Edition) – Since only ISBN functionally determines BookTitle, Publisher, and Edition, this combination would experience data redundancy if there were multiple authors for a single ISBN/Book.
* Author\_Num 🡪 LastName – Since only Author\_Num functionally determines LastName this combination would be repeated for each ISBN attribute that pertains to this partial dependency set since an Author can write one to many ISBNs/books and each book can have multiple editions.

**Transitive dependencies:**

* BookTitle 🡪 Publisher – Since the BookTitle functionally determines the Publisher, this combination would be redundant for each instance in which a new edition/version of a book requires a new ISBN but keeps the same title and publisher.

Example of undesirable dependencies:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ISBN** | **BookTitle** | **Author\_Num** | **LastName** | **Publisher** | **Royalty** | **Edition** |
| 110 | Birds 2.0 | 32 | Brooks | BBC LLC | $150.00 | 1 |
| 110 | BirdS 2.0 | 34 | Allen | BBC LLC | $170.00 | 1 |
| 123 | Nocturnal | 19 | Smithson | Gates LLP | $240.00 | 1 |
| 124 | Nocturnal | 34 | Allen | Gates LLP | $260.00 | 2 |
| 231 | Dream Big | 41 | Newman | ABC LLLP | $12320.92 | 1 |
| 235 | Dream Big | 32 | Brooks | ABC LLLP | $3123.10 | 2 |

These dependencies are undesirable because they cause data redundancy. This can lead to data inconstancies via update, insertion, and deletion anomalies.

**Part two: Show the 2NF database. – 1NF and no partial dependencies**

* **Table 1:** ISBN 🡪 (BookTitle, Publisher, Edition)
* **Table 2:** Author\_Num 🡪 LastName
* **Table 3:** (ISBN, Author\_Num) 🡪 Royalty

**Part three: Show the 3NF database.-2NF and no transitive dependencies**

* **Table 1:** ISBN 🡪 (BookTitle, Edition)
* **Table 2:** Author\_Num 🡪 LastName
* **Table 3:** (ISBN, Author\_Num) 🡪 Royalty
* **Table 4:** BookTitle 🡪 Publisher

