**Requirement 3** - The system must be able to add new inventory items (products) with no duplicate records.

The following code creates a stored procedure to add new items. It validates that the product does already exist in the system and the product does have a valid type.

drop procedure sp\_add\_product

go

create procedure sp\_add\_product (@pro\_name varchar(30), @pro\_instock int, @ty\_id int, @pro\_base numeric(5,2)) as

begin

--Checks for duplicate product names

if exists (select \* from t\_product where upper(pro\_name)=upper(@pro\_name))

begin

select @pro\_name, 'already exsists in this system as a product.'

return

end

else

--NEED Checks that type is valid

if not exists (select \* from t\_type where upper(ty\_id)=upper(@ty\_id))

begin

select @pro\_name, 'was not given a valid type.'

return

end

--inserts product data into t\_product table

begin transaction

insert into t\_product

(pro\_name, pro\_instock, ty\_id, pro\_base)

values

(@pro\_name, @pro\_instock, @ty\_id, @pro\_base)

if @@error<>0

begin

rollback transaction

select 'Product ', @pro\_name, ' not added'

return

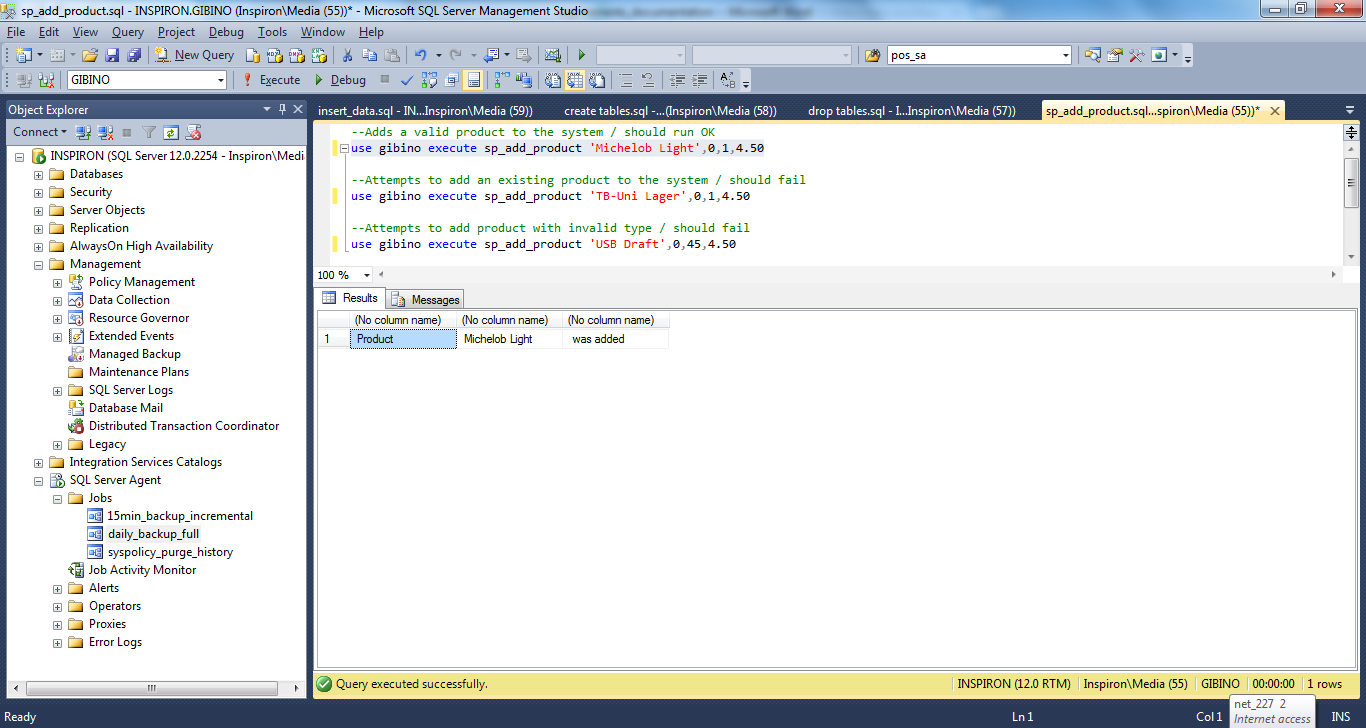
end

commit transaction

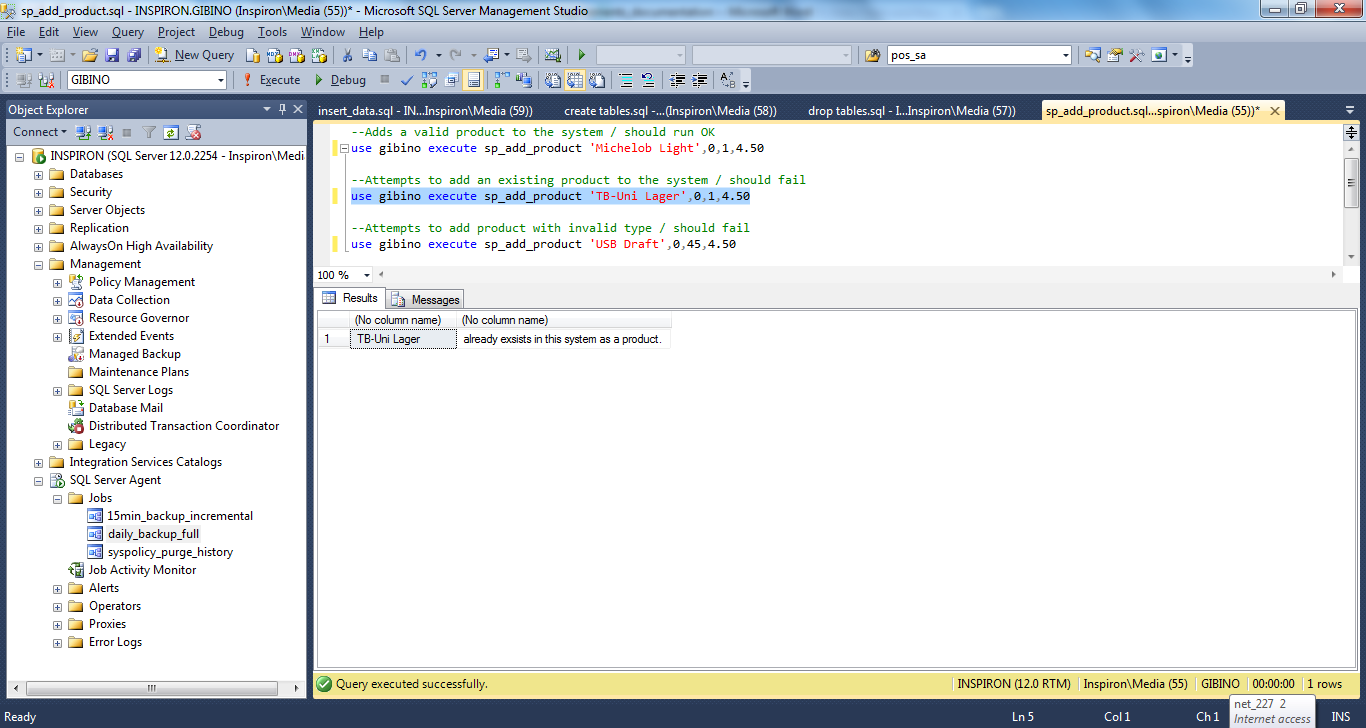
select 'Product ', @pro\_name, ' was added'

end

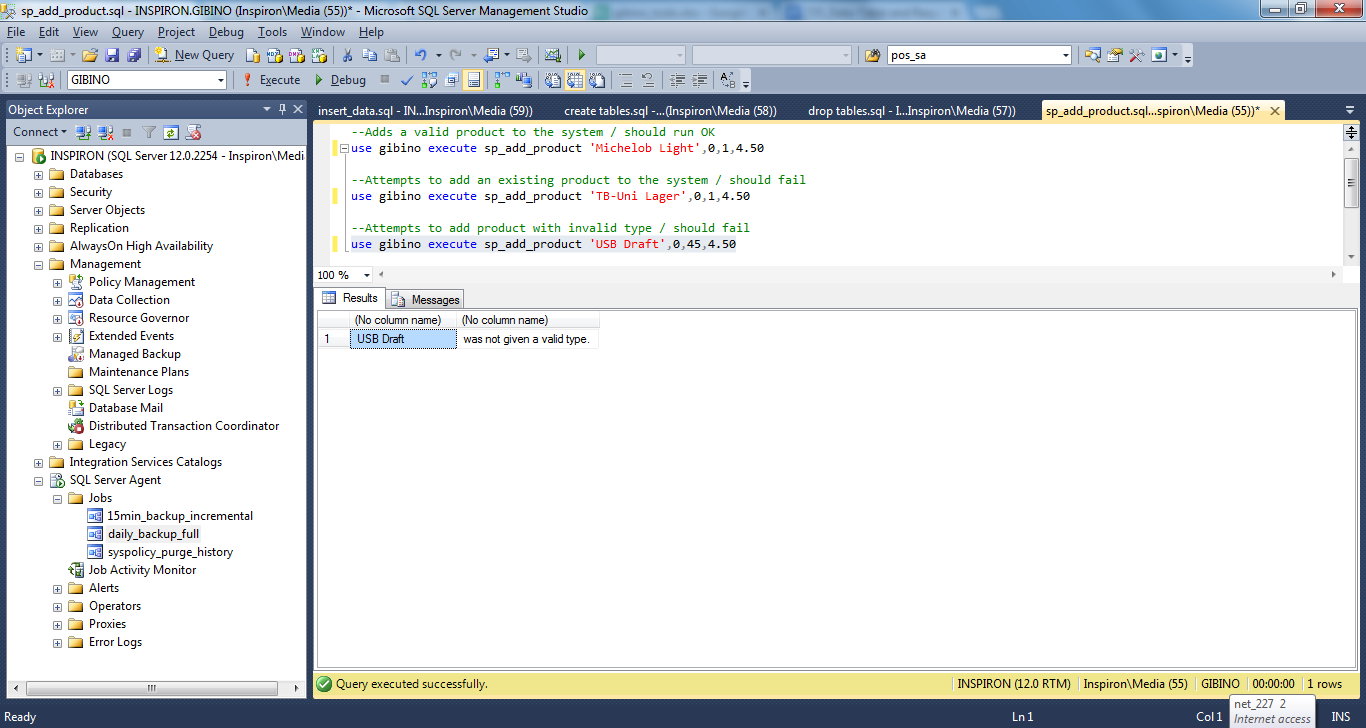
This screen shot demonstrates the insertion of a successful product.



This screen shot demonstrates a failed attempt to insert an existing product.



This screen shot demonstrates a failed attempt to insert a product with an invalid type.



**Requirement 7** – The system must display original prices and current prices

**Requirement 8** – The system must display % difference in price from current to original.

**Requirement 9** – The system must display daily high price for beer.

**Requirement 10** – The system must display daily low price for beer.

The following code creates a stored procedure view the Price Report data with a date parameter.

IF OBJECTPROPERTY(object\_id('dbo.sp\_price\_report'), N'IsProcedure') = 1

DROP PROCEDURE [dbo].[sp\_price\_report]

GO

CREATE PROCEDURE dbo.sp\_price\_report

@date datetime

AS

BEGIN

BEGIN TRANSACTION

IF OBJECT\_ID('dbo.t\_price\_diff', 'U') IS NOT NULL

drop table dbo.t\_price\_diff

create table dbo.t\_price\_diff

(

diff\_id int identity(1,1) primary key,

pro\_id int foreign key references dbo.t\_product(pro\_id),

diff\_perc decimal(5,1) not null

);

INSERT INTO T\_PRICE\_DIFF (PRO\_ID, DIFF\_PERC)

SELECT pri.pro\_id, ((pri.pro\_price - pro.pro\_base)/pro.pro\_base) \* 100

FROM t\_price pri inner join t\_product pro

on pri.pro\_id = pro.pro\_id;

if @@error <> 0

begin

rollback transaction

select ' There was a problem creating the price report'

return

end

commit transaction

select pro.pro\_name as Product, pro.pro\_base as OriginalPrice, pri.pro\_price as CurrentPrice, d.diff\_perc as PercentageDifference

, (CASE WHEN max(s.pro\_price) < pri.pro\_price THEN pri.pro\_price ELSE max(s.pro\_price) END) as DailyHigh

, min(s.pro\_price) as DailyLow

from t\_price pri

inner join t\_product pro on pri.pro\_id = pro.pro\_id

inner join t\_price\_diff d on d.pro\_id = pri.pro\_id

inner join t\_pos\_sales s on s.pro\_id = d.pro\_id

where DAY(s.pos\_datetime) = DAY(@date)

group by pri.pro\_id, pro.pro\_name,pro.pro\_base, pri.pro\_price, d.diff\_perc

UNION

select pro.pro\_name as Product, pro.pro\_base as OriginalPrice, pri.pro\_price as CurrentPrice, 0 as PercentageDifference, pro.pro\_base as DailyHigh, pro.pro\_base as DailyLow

from t\_price pri

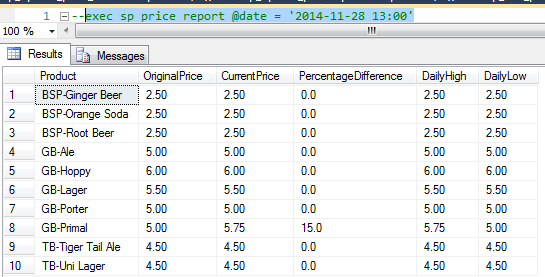
inner join t\_product pro on pri.pro\_id = pro.pro\_id

inner join t\_price\_diff d on d.pro\_id = pri.pro\_id

where pri.pro\_id not in (select pro\_id from t\_pos\_sales where DAY(pos\_datetime) = DAY(@date));

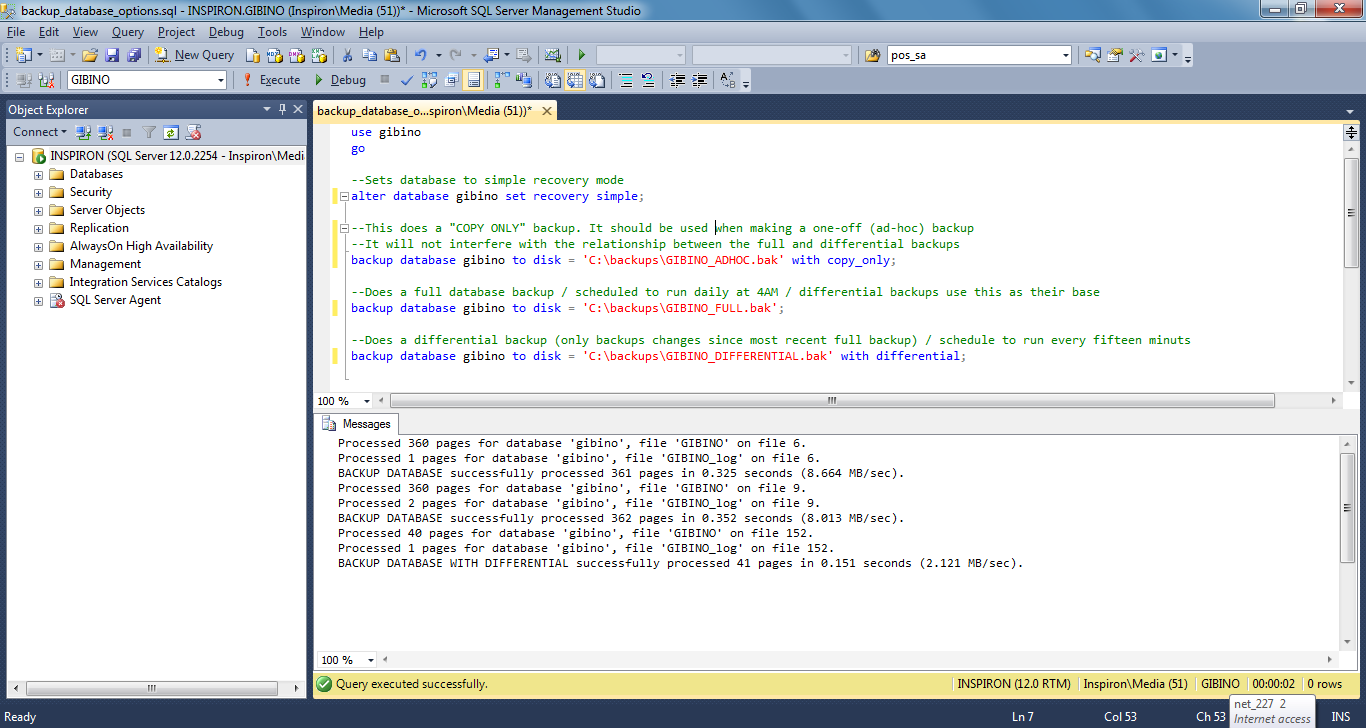
END

This screen capture demonstrates successful generation of a Price Report for the day input.

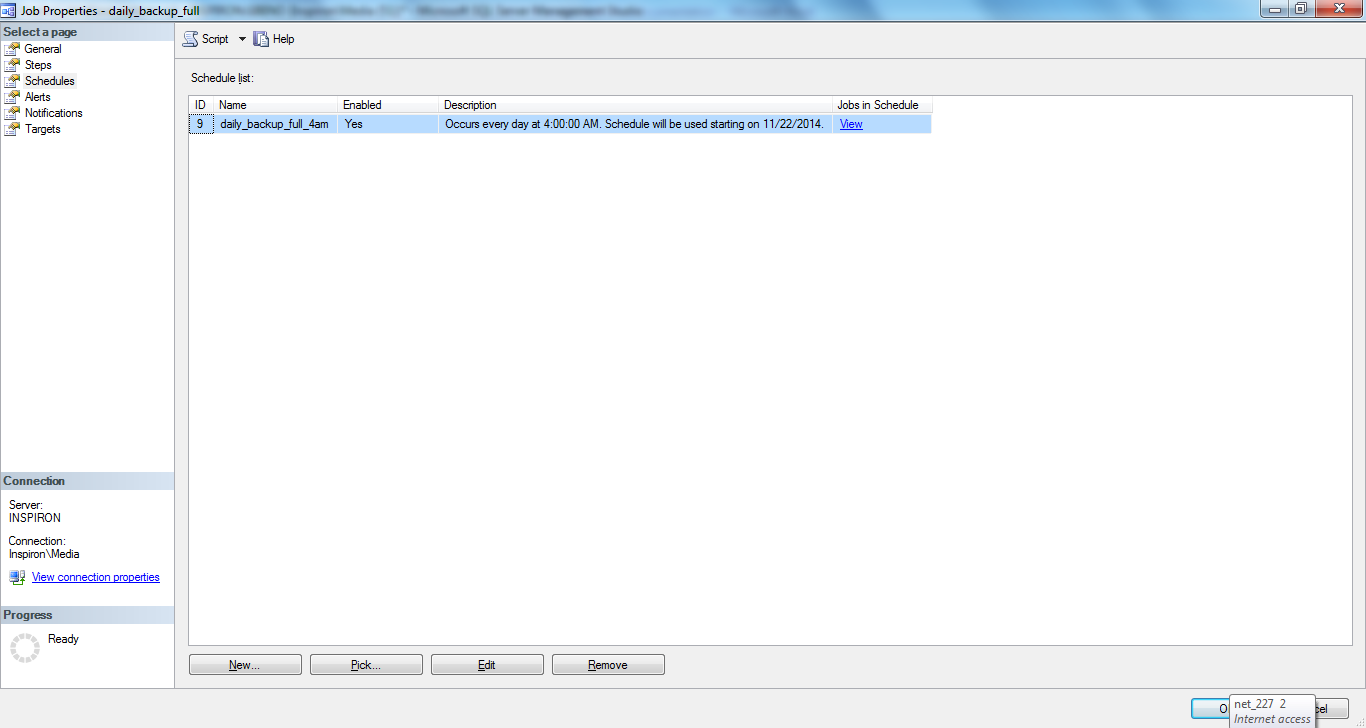


**Requirement 14** -The system must create a nightly backup, after normal business hours.

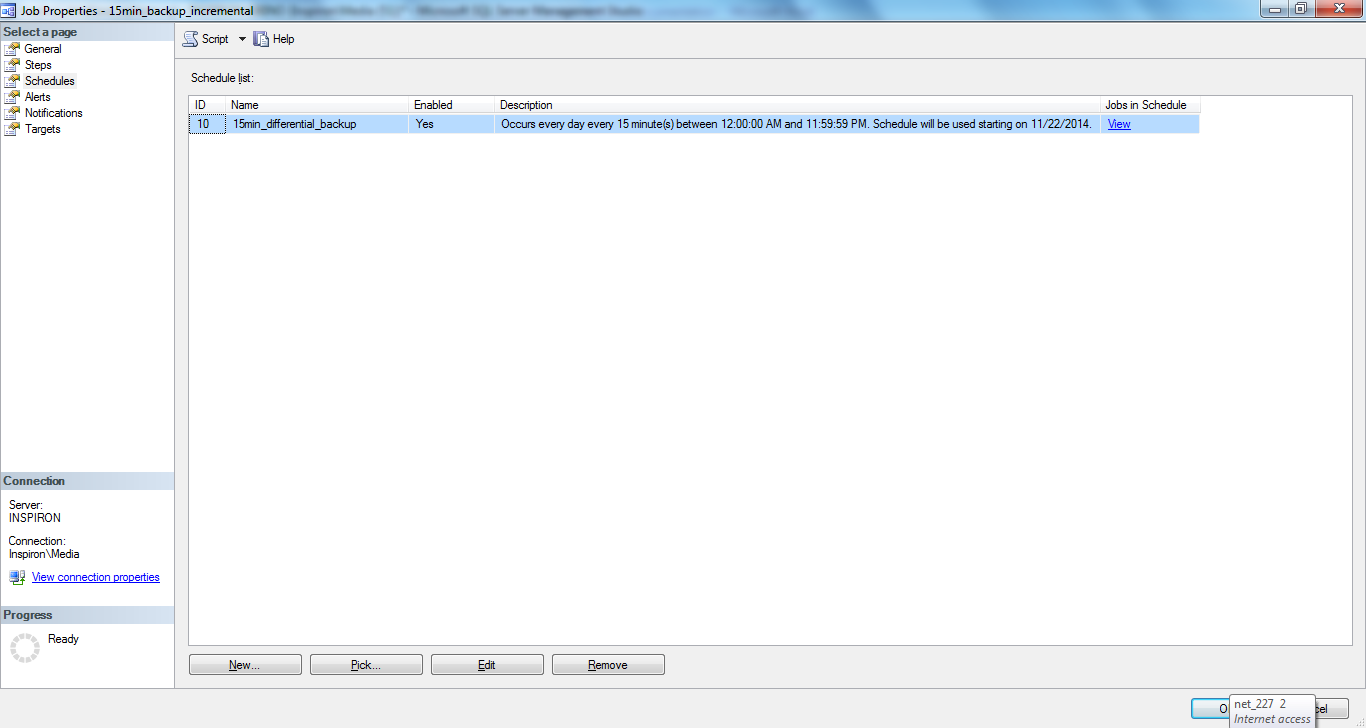
The following code demonstrates how to setup the database for ad hoc, full, and differential backups. It also shows how to put the database into simple recovery mode.



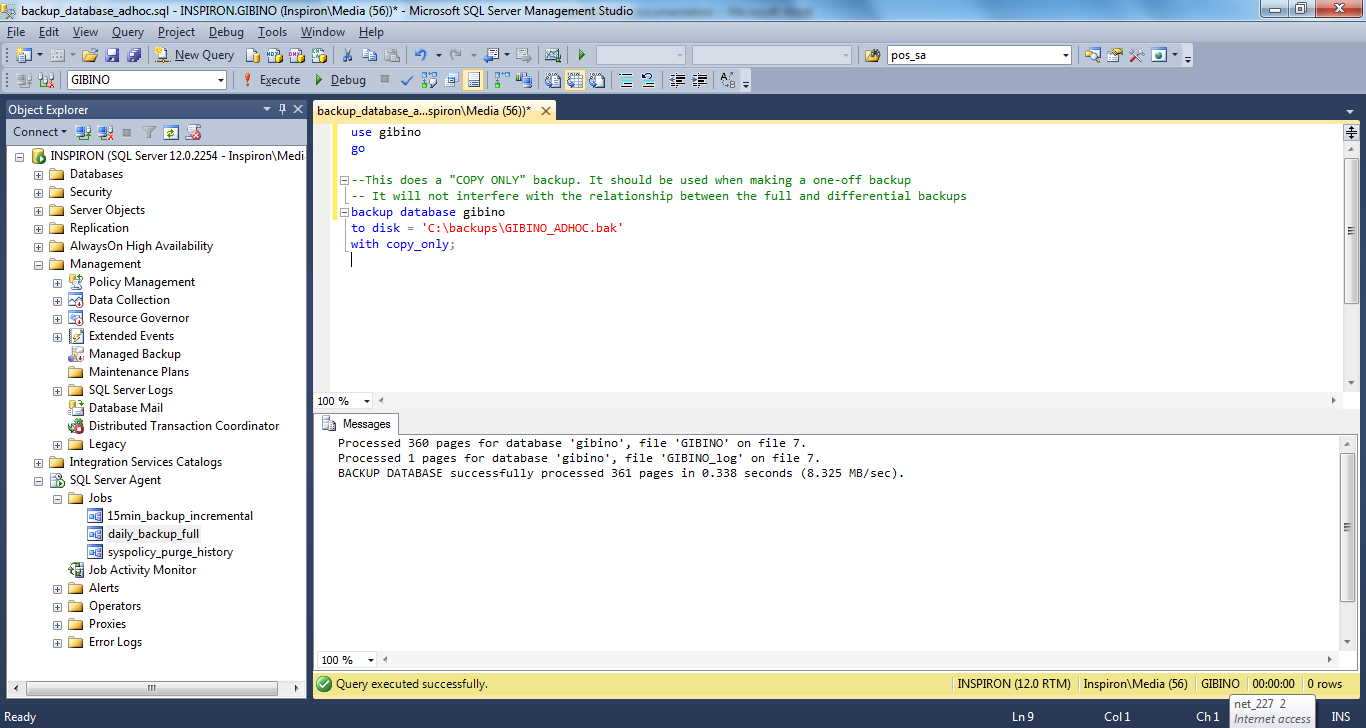
This screen capture shows that the full backup is a scheduled job, which runs at 4:00 AM every day.



This screen capture shows that the differential backup is a scheduled job, which runs every 15 minutes.



This screen capture demonstrates the code and output for a successful ad hoc backup.



**Requirement 21** – The system must be able to display the unpaid items for a customer.

The following code creates a stored procedure to query the system for unpaid items. It validates that the product does already exist in the system and the product does have a valid type.

IF OBJECTPROPERTY(object\_id('sp\_customer\_tab'), N'IsProcedure') = 1

DROP PROCEDURE [dbo].[sp\_customer\_tab]

GO

CREATE PROCEDURE dbo.sp\_customer\_tab

@cus\_id INT

As

Begin

If not exists (Select \* from T\_CUSTOMER where cus\_id = @cus\_id)

begin

select 'Invalid Customer!'

return

end

Select pro.pro\_name, s.pro\_price, s.pos\_qty

from t\_customer c

inner join t\_pos\_sales s on c.cus\_id=s.cus\_id

inner join t\_product pro on pro.pro\_id=s.pro\_id

where s.pos\_paid=0

and s.cus\_id = @cus\_id

Select c.cus\_fname, c.cus\_lname,sum(s.pro\_price) As SubTotal

from t\_customer c

inner join t\_pos\_sales s on c.cus\_id=s.cus\_id

inner join t\_product pro on pro.pro\_id=s.pro\_id

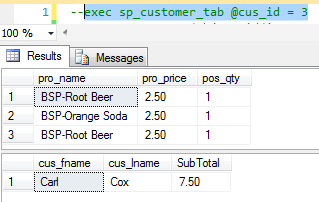
where s.pos\_paid=0

and s.cus\_id = @cus\_id

group by c.cus\_id,c.cus\_fname, c.cus\_lname

End

This screen capture demonstrates an execution of stored procedure for an existing customer.



This screen capture demonstrates the validation for an existing customer.

