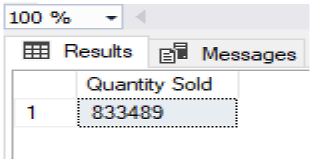
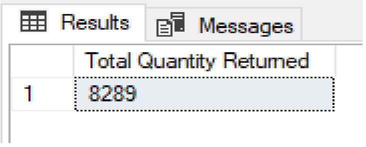
```sql
use Commercial_Project

-- Pull the sum of quantity from each data table. Total Quantity Sold = 833,489 and Total Quantity
Returned = 8,289

select SUM(quantity) AS 'Quantity Sold' from Transaction_Data

select SUM(quantity) AS 'Total Quantity Returned' from Return_Data
```

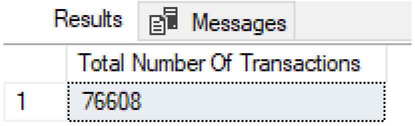


```
  Quantity Sold
1 833489
```



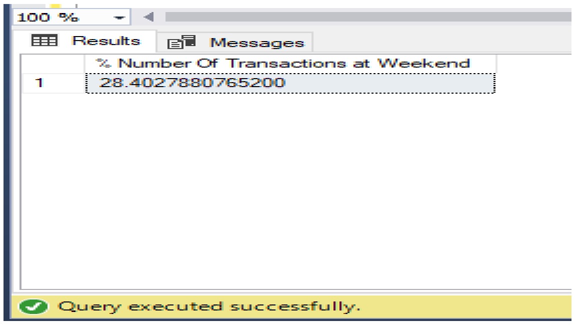
```
  Total Quantity Returned
1 8289
```

```sql
-- Pull the 'Total Transactions' and 'Total Returns' to calculate the count of rows from each data
table. 269,720 transactions and 7,087 returns

select COUNT(*) AS 'Total Transactions' from Transaction_Data

select COUNT(return_date) AS 'Total Returns' from Return_Data

-- Pull the 'Return Rate' to calculate the ratio of quantity returned to quantity sold (format as %).
Overall return rate of 0.99%

select (SUM(return_data.quantity) / (select SUM(transaction_data.quantity) from Transaction_Data)) *
100 as result from Return_Data
left join Calendar on Return_Data.return_date = Calendar.Date
inner join Transaction_Data on Transaction_Data.transaction_date = Calendar.Date

select cast(833489 / 7223 as decimal(10,2)) as result  * CAST(100 as decimal(10,2))) as result

select 8289/8290 as result

-- Pull the total number of transactions on weekends. 76,608 total weekend transactions

select count(t.quantity) as 'Total Number Of Transactions' from Transaction_Data t
left join Calendar c on c.Date = t.transaction_date
where c.Weekend = 'Y'
```



```
  Total Number Of Transactions
1 76608
```

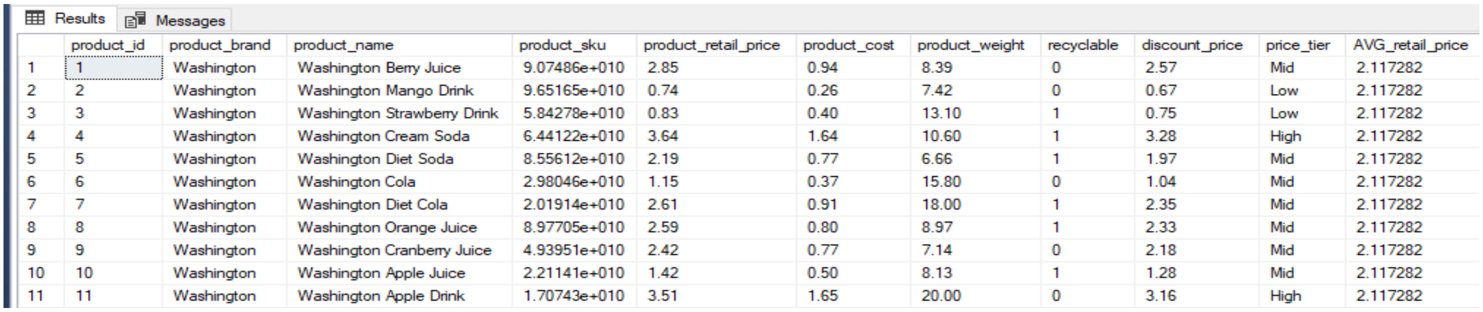
```sql
-- Create a new measure named "% Weekend Transactions" to calculate weekend transactions as a
percentage of total transactions (format as %).
-- I should see 28.4% weekend transactions

select CAST(COUNT(transaction_date) as decimal(10,2)) / (select CAST(COUNT(transaction_date) AS
decimal(10,2)) from Transaction_Data) * 100 AS '% Number Of Transactions at Weekend' from
Transaction_Data
inner join calendar on calendar.date = Transaction_Data.transaction_date
where Calendar.weekend = 'Y'
```



	% Number Of Transactions at Weekend
1	28.4027880765200

Query executed successfully.

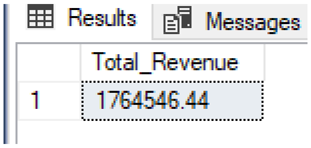
```
-- Calculate % Total Count of Returns and % Of Total Count of Transactions

select cast(COUNT(return_date) as decimal(10,2)) / (select cast(COUNT(return_date) as decimal(10,2))
from Return_Data) * 100 from Return_Data

select cast(COUNT(transaction_date) as decimal(10,2)) / (select CAST(COUNT(transaction_date) AS
decimal(10,2)) from Transaction_Data) * 100 from Transaction_Data

-- Create a new measure to calculate the AVG Retail Price

select CAST(AVG(product_retail_price) AS decimal(10,2)) AS AVG_retail_price from Products

-- OR I could have used the OVER() clause Window function to calculate the AVG Retail_Price for all
company

select *, AVG(product_retail_price) OVER() as AVG_retail_price from Products
```



	product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight	recyclable	discount_price	price_tier	AVG_retail_price
1	1	Washington	Washington Berry Juice	9.07486e+010	2.85	0.94	8.39	0	2.57	Mid	2.117282
2	2	Washington	Washington Mango Drink	9.65165e+010	0.74	0.26	7.42	0	0.67	Low	2.117282
3	3	Washington	Washington Strawberry Drink	5.84278e+010	0.83	0.40	13.10	1	0.75	Low	2.117282
4	4	Washington	Washington Cream Soda	6.44122e+010	3.64	1.64	10.60	1	3.28	High	2.117282
5	5	Washington	Washington Diet Soda	8.55612e+010	2.19	0.77	6.66	1	1.97	Mid	2.117282
6	6	Washington	Washington Cola	2.98046e+010	1.15	0.37	15.80	0	1.04	Mid	2.117282
7	7	Washington	Washington Diet Cola	2.01914e+010	2.61	0.91	18.00	1	2.35	Mid	2.117282
8	8	Washington	Washington Orange Juice	8.97705e+010	2.59	0.80	8.97	1	2.33	Mid	2.117282
9	9	Washington	Washington Cranberry Juice	4.93951e+010	2.42	0.77	7.14	0	2.18	Mid	2.117282
10	10	Washington	Washington Apple Juice	2.21141e+010	1.42	0.50	8.13	1	1.28	Mid	2.117282
11	11	Washington	Washington Apple Drink	1.70743e+010	3.51	1.65	20.00	0	3.16	High	2.117282

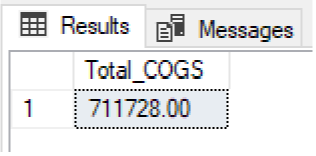
```
-- Calculate the Total Count of Transactions, but only for products > than the average price.
-- So, I want to see the # of transactions  for product prices higher than Overall AVG priced items.

-- SCALAR UNCORRELATED SUBQUERY IN A WHERE CLAUSE

select * from Transaction_Data
select * from products

select COUNT(transaction_date) AS number_of_transactions from Transaction_Data
right join Products on products.product_id = Transaction_Data.product_id
WHERE products.product_retail_price >=


(select CAST(AVG(products.product_retail_price) AS decimal(10,2)) AS avg_retail_price from Products)


-- Calculate 'Total Revenue' based on transaction quantity and product retail price. I should see a
total revenue of $1,764,546

select ROUND(SUM(t.quantity * p.product_retail_price),0) from Transaction_Data t
inner join Products p on p.product_id = t.product_id

-- I can also add a new column called Revenue to the Transaction_data table and update it multiplying
quantity by retail price
```

```
ALTER TABLE Transaction_data
ADD Revenue DECIMAL(10,2)

UPDATE Transaction_Data
SET Revenue = t.quantity * p.product_retail_price from Transaction_Data t
inner join Products p on p.product_id = t.product_id

select SUM(revenue) AS Total_Revenue from Transaction_Data
```



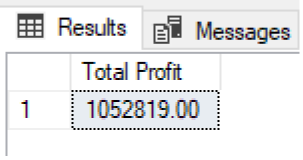
```
   Total_Revenue
1  1764546.44
```

```
-- Calculate 'Total COGS' based on transaction quantity and product cost, and format. Spot check: I
should see a total cost of $711,728

select ROUND(SUM(t.quantity * p.product_cost),0) from Transaction_Data t
inner join Products p on p.product_id = t.product_id

-- I can also add a new column called COGS and update it multiplying quantity by product cost
ALTER TABLE Transaction_data
ADD COGS DECIMAL(10,2)

UPDATE Transaction_Data
SET COGS = t.quantity * p.product_cost from Transaction_Data t
inner join Products p on p.product_id = t.product_id

select ROUND(SUM(COGS),0) AS Total_COGS from Transaction_Data
```



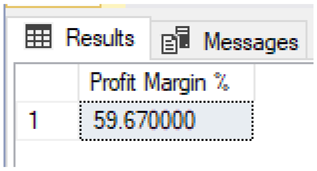
```
   Total_COGS
1  711728.00
```

```
-- Calculate 'Total Profit' to calculate total revenue minus total cost, and format as $. Spot check: I
should see a total profit of $1,052,819

select ROUND(SUM(t.quantity * p.product_retail_price) - SUM(t.quantity * p.product_cost),0) from
Transaction_Data t
inner join Products p on p.product_id = t.product_id
-- OR
select ROUND(SUM(Revenue) - SUM(COGS),0) AS 'Total_Profit' from Transaction_Data

-- I can also add a new column called Profit and update it subtracting COGS from Revenue

select * from Transaction_Data

ALTER TABLE transaction_data
ADD Profit decimal(10,2)

UPDATE Transaction_Data
SET Profit = (Revenue - COGS)

select ROUND(SUM(Profit),0) AS 'Total Profit' from Transaction_Data
```



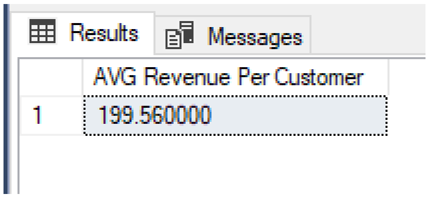
	Total Profit
1	1052819.00

```sql
-- Calculate "Profit Margin" by dividing total profit by total revenue (format as %). Spot check: You should see an overall profit margin of 59.67%

select ROUND(SUM(Profit) / SUM(Revenue) * 100,2) AS 'Profit Margin %' from Transaction_Data
```



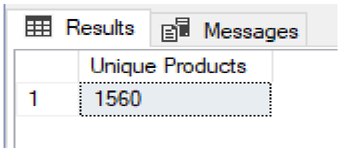
	Profit Margin %
1	59.670000

```sql
-- Report the ‘AVG Revenue Per Sales Customer'

select ROUND(SUM(Revenue)  /  COUNT(DISTINCT customer_id),2) AS 'AVG Revenue Per Customer' from
Transaction_Data
```



	AVG Revenue Per Customer
1	199.560000

```sql
-- Calculate the number of unique product names in the Products table. Spot check: I should see 1,560 unique products

select COUNT(DISTINCT product_name) AS 'Unique Products' from Products
```



	Unique Products
1	1560