[redacted] Manipol, Marion Jose S [redacted]

Activities of HO2: Generating Reports from a Data Warehouse

The report begins with an overview of the assignment objectives. For HO2, CSV files were exported from the dataset output in HO1 and imported into (a) Microsoft Excel and (b) a local MySQL database. Two distinct methods, “X” and SQL queries within the MySQL database, were employed to generate reports as outlined in the provided list. The purpose of this assignment is to offer hands-on experience in report generation and data visualization while comparing the effectiveness of SQL queries used in OLAP operations with the capabilities offered by existing spreadsheet tools.

Report #1: Average monthly sales, in terms of quantity sold, per product line

1. Import the relevant tables from our data warehouse into sheets (the fact table, and the product dimension)
2. Using VLOOKUP, get the productLine of all products from the dimension table. After extracting the month and year from the orderDate column from the fact table, this gives you 3 new columns.
3. Select all data within the fact table, including the 3 new columns, and create a pivot table.
4. In the pivot table, have the productLine as the column. The month and year will be the rows. The quantity will be the value.
5. The pivot table will display the quantity sold in total by default, but this can be changed to display the average instead.

Spreadsheet Instructions

```sql
SELECT
   dp.productLine,
   EXTRACT(YEAR FROM fo.orderDate) AS year,
   EXTRACT(MONTH FROM fo.orderDate) AS month,
   AVG(fo.quantity) AS avg_quantity_sold
FROM
   fact_orders fo
JOIN
   dim_product dp ON fo.productKey = dp.productKey
GROUP BY
   dp.productLine,
   EXTRACT(YEAR FROM fo.orderDate),
   EXTRACT(MONTH FROM fo.orderDate)
ORDER BY
   dp.productLine,
   year,
   month,
   avg_quantity_sold;
```

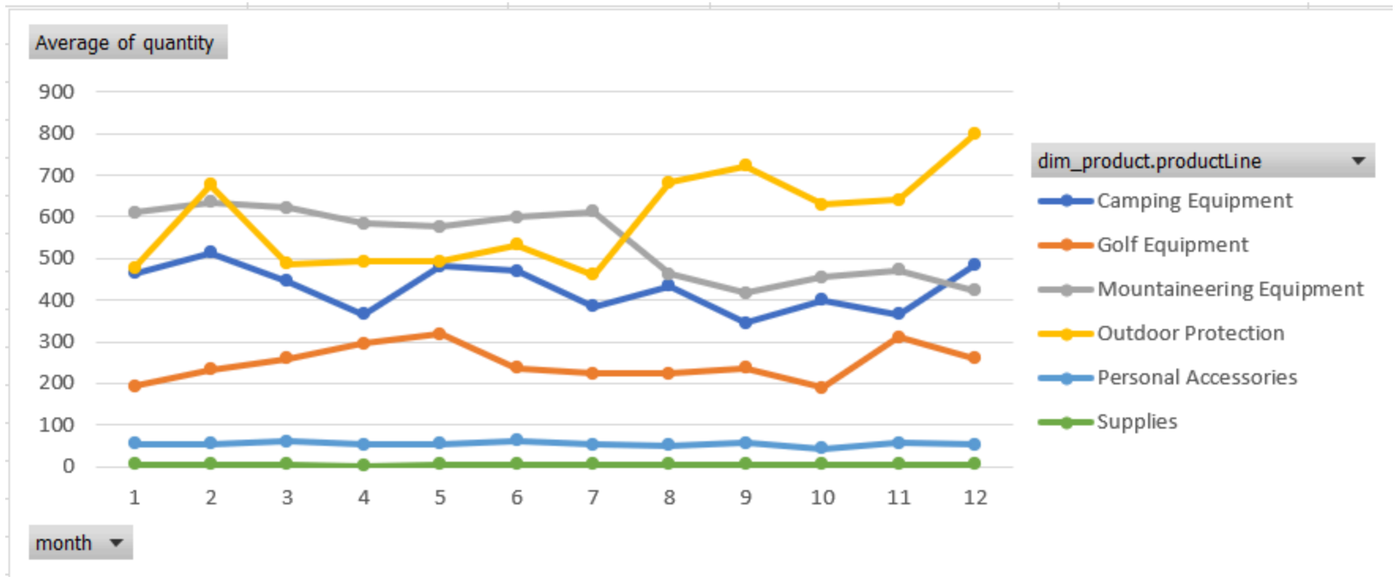
SQL Query

Average of quantity	Column Labels				
Row Labels	Camping Equipment	Golf Equipment	Mountaineering Equipment	Outdoor Protection	Personal Accesso
October					
November					
December					
2015	**387.770846**	**208.4187528**		**970.0267857**	**68.71079**
January	360.0522088	236.9637306		949.2380952	72.57783
February	418.9166667	210.9866667		1149.230769	63.54209
March	345.2708333	167.2995781		890.2207792	79.76455
April	349.375817	201.9243243		775.3493151	65.43252
May	370.579288	236.8343195		917.1011236	73.88319
June	480.0106007	195.8111111		1005.457364	83.27327
July	332.9360465	166.9927007		931.2745098	73.39450
August	459.7292994	196.9836066		1029.691667	62.08465
September	360.1449275	192.8571429		949.4935065	70.55635
October	347.143617	182.7926829		890.6721311	49.09780

Spreadsheet Results

	productLine	year	month	avg_quantity_sold
1	Camping Equipment	2015	1	360.0522
2	Camping Equipment	2015	2	418.9167
3	Camping Equipment	2015	3	345.2708
4	Camping Equipment	2015	4	349.3758
5	Camping Equipment	2015	5	370.5793
6	Camping Equipment	2015	6	480.0106
7	Camping Equipment	2015	7	332.9360
8	Camping Equipment	2015	8	459.7293
9	Camping Equipment	2015	9	360.1449
10	Camping Equipment	2015	10	347.1436
11	Camping Equipment	2015	11	330.3263
12	Camping Equipment	2015	12	455.6555
13	Camping Equipment	2016	1	506.3281
14	Camping Equipment	2016	2	472.7344
15	Camping Equipment	2016	3	481.8254
16	Camping Equipment	2016	4	369.5418
17	Camping Equipment	2016	5	351.9177
18	Camping Equipment	2016	6	312.0809
19	Camping Equipment	2016	7	267.7406
20	Camping Equipment	2016	8	385.6564

SQL Query Results



Visualization

>> Refine the query to show the average monthly sales for each brand in a product line

1. Import the relevant tables from our data warehouse into sheets (the fact table, and the product dimension)
2. Using VLOOKUP, get the productLine and productBrand of all products from the dimension table. After extracting the month and year from the orderDate column from the fact table, this gives you 4 new columns.
3. Select all data within the fact table, including the 4 new columns, and create a pivot table.
4. In the pivot table, have the productLine and productBrand as the columns. The month and year will be the rows. The value will be the grossSale =quantity*unitPrice.
5. The pivot table will display the total monthly sales by default, but this can be changed to display the average instead.

Spreadsheet Instructions

```sql
SELECT
 dp.productLine AS product_line,
 dp.productBrand,
 EXTRACT(YEAR FROM fo.orderDate) AS year,
 EXTRACT(MONTH FROM fo.orderDate) AS month,
 AVG(fo.quantity * dp.unitPrice) AS avg_monthly_sales
FROM
 fact_orders fo
JOIN
 dim_product dp ON fo.productKey = dp.productKey
GROUP BY
 dp.productLine,
 dp.productBrand,
 EXTRACT(YEAR FROM fo.orderDate),
 EXTRACT(MONTH FROM fo.orderDate)
ORDER BY
 dp.productLine,
 dp.productBrand,
 year,
 month;
```

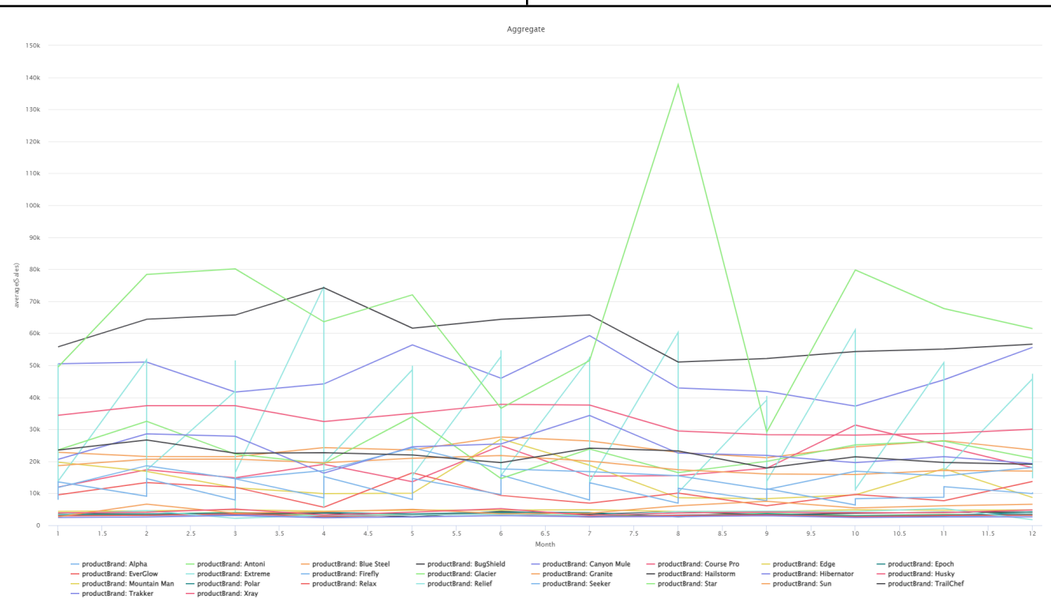
SQL Query

Average of grossSale	Column Labels							
	Camping Equipment							Camping Equipment Total
Row Labels	Canyon Mule	EverGlow	Extreme	Firefly	Hibernator	Star	TrailChef	
October								
November								
December								
2015	**40496.17239**	**10128.39076**	**40470.38262**	**8117.233587**	**18522.82217**	**55161.10537**	**19121.13827**	**25365.14763**
January	59967.11214	7162.664762	25702.1	7783.531395	12058.58	41019.50634	16761.41057	24922.41775
February	34360.17327	12129.14353	48789.99667	8680.0614	18868.22324	61624.72977	23316.72227	26914.5358
March	30858.25739	7485.75	49380.28857	11165.22826	14949.52714	81567.82313	18044.27381	32571.20597
April	36626.118	4523.576286	42211.86667	9517.550435	11370.93151	54940.19548	19711.55696	21995.02428
May	34161.76439	12721.07696	33918.23333	6388.052889	16465.53091	55475.51956	18243.81404	24193.51942
June	42257.04529	12741.49347	21349.788	9395.168947	21029.46178	34063.6184	18189.19911	23714.58837
July	65632.34087	10377.65625	31045.775	5817.639778	32717.37462	58994.15382	22598.0895	31942.79703
August	33900.67359	10908.83725	54815.9025	6813.719111	20032.40019	118938.597	25659.34521	29210.8451
September	34444.53224	8110.217455	29014.45333	7863.951	22136.89118	34901.74486	15780.13784	20343.38974
October	37971.4375	9280.735429	42329.36857	6773.730385	20590.7575	66698.52765	17276.80647	27711.63883
November	43686.41698	8491.4475	39412.76118	8320.872075	19594.05391	46371.90607	17979.71809	24584.92005
December	45988.27082	16604.30531	46403.275	9030.119455	16266.51451	39382.96324	17845.98984	24411.15843
2016	**43793.85202**	**7873.213449**	**46332.48905**	**7187.265388**	**20128.07264**	**59010.52173**	**20199.12669**	**27612.83833**
January	56414.11488	6223.58	27283.358	7151.956	16804.52333	53415.43794	27233.10065	28046.33466
February	57981.5169	9475.5996	46410.75043	8577.5642	27893.20264	71538.21982	30738.35281	35416.05993

Spreadsheet Results

	product_line	productBrand	year	month	avg_monthly_sales
1	Camping Equipment	Canyon Mule	2015	1	59967.112143
2	Camping Equipment	Canyon Mule	2015	2	34360.173265
3	Camping Equipment	Canyon Mule	2015	3	30858.257391
4	Camping Equipment	Canyon Mule	2015	4	36626.118000
5	Camping Equipment	Canyon Mule	2015	5	34161.764386
6	Camping Equipment	Canyon Mule	2015	6	42257.045294
7	Camping Equipment	Canyon Mule	2015	7	65632.340870
8	Camping Equipment	Canyon Mule	2015	8	33900.673594
9	Camping Equipment	Canyon Mule	2015	9	34444.532239
10	Camping Equipment	Canyon Mule	2015	10	37971.437500
11	Camping Equipment	Canyon Mule	2015	11	43686.416984
12	Camping Equipment	Canyon Mule	2015	12	45988.270816
13	Camping Equipment	Canyon Mule	2016	1	56414.114884
14	Camping Equipment	Canyon Mule	2016	2	57981.516905
15	Camping Equipment	Canyon Mule	2016	3	33014.940000
16	Camping Equipment	Canyon Mule	2016	4	39133.416212
17	Camping Equipment	Canyon Mule	2016	5	41532.165455
18	Camping Equipment	Canyon Mule	2016	6	32581.790000
19	Camping Equipment	Canyon Mule	2016	7	51947.734000

SQL Query Results



Visualization

>> Refine the query to show the total monthly profit per product line

1. Import the relevant tables from our data warehouse into sheets (the fact table, and the product dimension)
2. Using VLOOKUP, get the productLine, unitCost, and unitPrice of all products from the dimension table. Profit is calculated as =quantity*(unitPrice-unitCost). After extracting the month and year from the orderDate column from the fact table, this gives you 6 new columns.
3. Select all data within the fact table, including the 6 new columns, and create a pivot table.
4. In the pivot table, have the productLine as the columns. The month and year will be the rows. The profit will be the value.
5. The pivot table will display the total monthly profits.

Spreadsheet Instructions

```sql
SELECT
 dp.productLine AS product_line,
 EXTRACT(YEAR FROM fo.orderDate) AS year,
 EXTRACT(MONTH FROM fo.orderDate) AS month,
 SUM(fo.quantity * (dp.unitPrice - dp.unitCost)) AS total_monthly_profit
FROM
 fact_orders fo
JOIN
 dim_product dp ON fo.productKey = dp.productKey
GROUP BY
 dp.productLine,
 EXTRACT(YEAR FROM fo.orderDate),
 EXTRACT(MONTH FROM fo.orderDate)
ORDER BY
 dp.productLine,
 year,
 month;
```

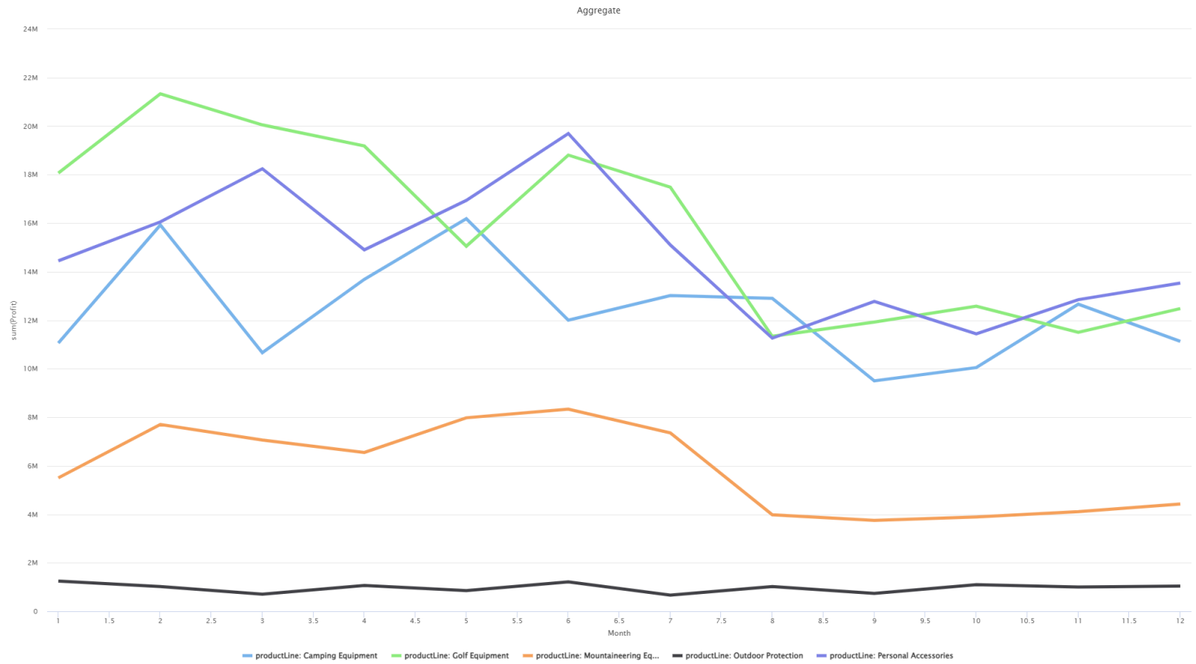
SQL Query

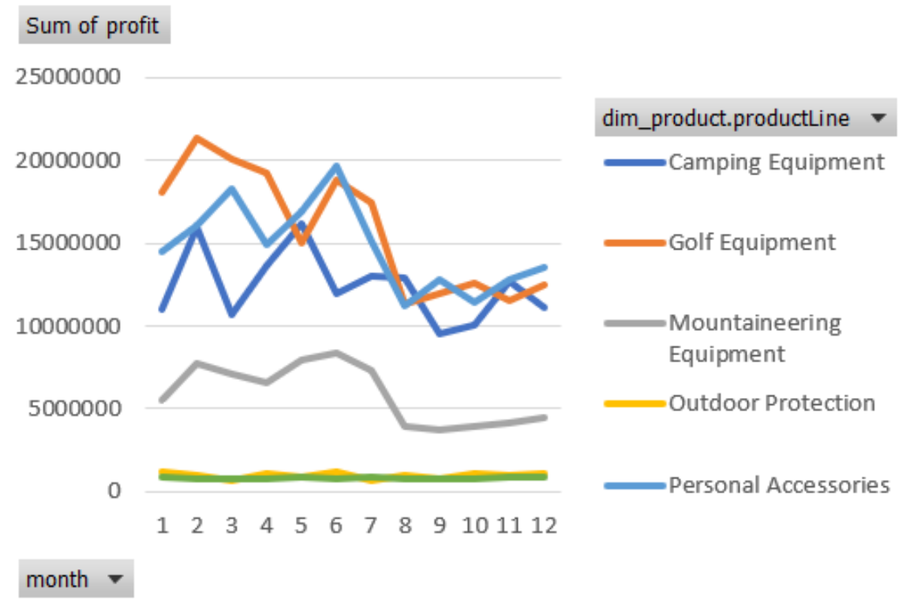
3	Sum of profit	Column Labels					
4	Row Labels	Camping Equipment	Golf Equipment	Mountaineering Equipment	Outdoor Protection	Personal Accessories	Supplies
30	October						0
31	November						0
32	December						0
33	**2015**	**34329808.79**	**37981782.2**		**5814178.61**	**37531854.91**	**0**
34	January	2616768.73	3387949.15		652611.09	2704206.21	0
35	February	3244434.46	3878231.36		560939.7	2699088.2	0
36	March	1757960.37	3274753.28		295974.83	3544320.44	0
37	April	2761397.4	2947050.42		559611.41	2390534.12	0
38	May	3063850.33	2759828.77		409837.55	3174093.58	0
39	June	2836932.61	3535707.58		684464.14	3638788.05	0
40	July	2169721.72	2897080.83		234998.09	3041075.03	0
41	August	3811316.21	3100422.03		607692.82	3213460.96	0
42	September	3028159.78	3066826.09		346909.09	3163339.48	0
43	October	2064444.26	3466888.09		499532.17	3210948.98	0
44	November	3840419.62	2638148.87		468161.84	3305929.3	0
45	December	3134403.3	3028895.73		493445.88	3446070.56	0
46	**2016**	**40411914.47**	**46394032.62**	**16098083.71**	**4052586.03**	**47487554.37**	**0**
47	January	3039283.6	4104718.95	1077411.82	381163.01	3593585.51	0

Spreadsheet Results

	product_line	year	month	total_monthly_profit
1	Camping Equipment	2015	1	2616768.73
2	Camping Equipment	2015	2	3244434.46
3	Camping Equipment	2015	3	1757960.37
4	Camping Equipment	2015	4	2761397.40
5	Camping Equipment	2015	5	3063850.33
6	Camping Equipment	2015	6	2836932.61
7	Camping Equipment	2015	7	2169721.72
8	Camping Equipment	2015	8	3811316.21
9	Camping Equipment	2015	9	3028159.78
10	Camping Equipment	2015	10	2064444.26
11	Camping Equipment	2015	11	3840419.62
12	Camping Equipment	2015	12	3134403.30
13	Camping Equipment	2016	1	3039283.60
14	Camping Equipment	2016	2	4455734.41

SQL Query Results





Visualization

Report 1 Comparison and OLAP Analysis

As seen in the steps above, the usage of pivot tables for Excel and the usage of GROUP BY in SQL can be an application of the **"roll-up**" operation in OLAP. We rolled up the category (from products themselves to product lines as a whole) going up the hierarchy of the dimension "product" from "Product Brand" (Canyon Mule, EverGlow, Firefly, etc.) to "Product Line" (Camping Equipment, Golf Equipment, etc.). The hierarchy of the Product dimension is as follows: Product Name, Product Brand, and Product Line.

As for the differences in approach, in the Excel approach, we had to import ONLY the important tables into our spreadsheets and create a pivot table from that, while the SQL query had everything already loaded on our MySQL Workbench data warehouse, and we only had to filter which dimensions/tables we would take from for generating our report using the 'FROM' clause.

Report #2: Performance of the brands per country in terms of gross sales and net profit

1. Import into separate sheets the fact table, the product dimension, and the retailer dimension.
2. Use VLOOKUP to retrieve the product.productBrand, product.unitCost, product.unitPrice, and retailer.country from the dimension sheets. Gross sales are calculated as **=quantity*unitPrice**. Profit is calculated as **=quantity*(unitPrice-unitCost)**. This will yield 6 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: productBrand
    b. Column: country
    c. Value: grossSales, profit
5. The pivot table should display the total gross sales and net profit for each brand per country.

Spreadsheet Instructions

```sql
SELECT
 dp.productBrand AS Brand,
 dr.country AS Country,
 SUM(fo.quantity * dp.unitPrice) AS GrossSales,
 SUM(fo.quantity * (dp.unitPrice - dp.unitCost)) AS NetProfit
FROM
 fact_orders fo
JOIN
 dim_product dp ON fo.productKey = dp.productKey
JOIN
 dim_retailers dr ON fo.retailerKey = dr.retailerKey
GROUP BY
 dp.productBrand, dr.country
ORDER BY
 Brand,
 dr.country, NetProfit DESC
```

SQL Query

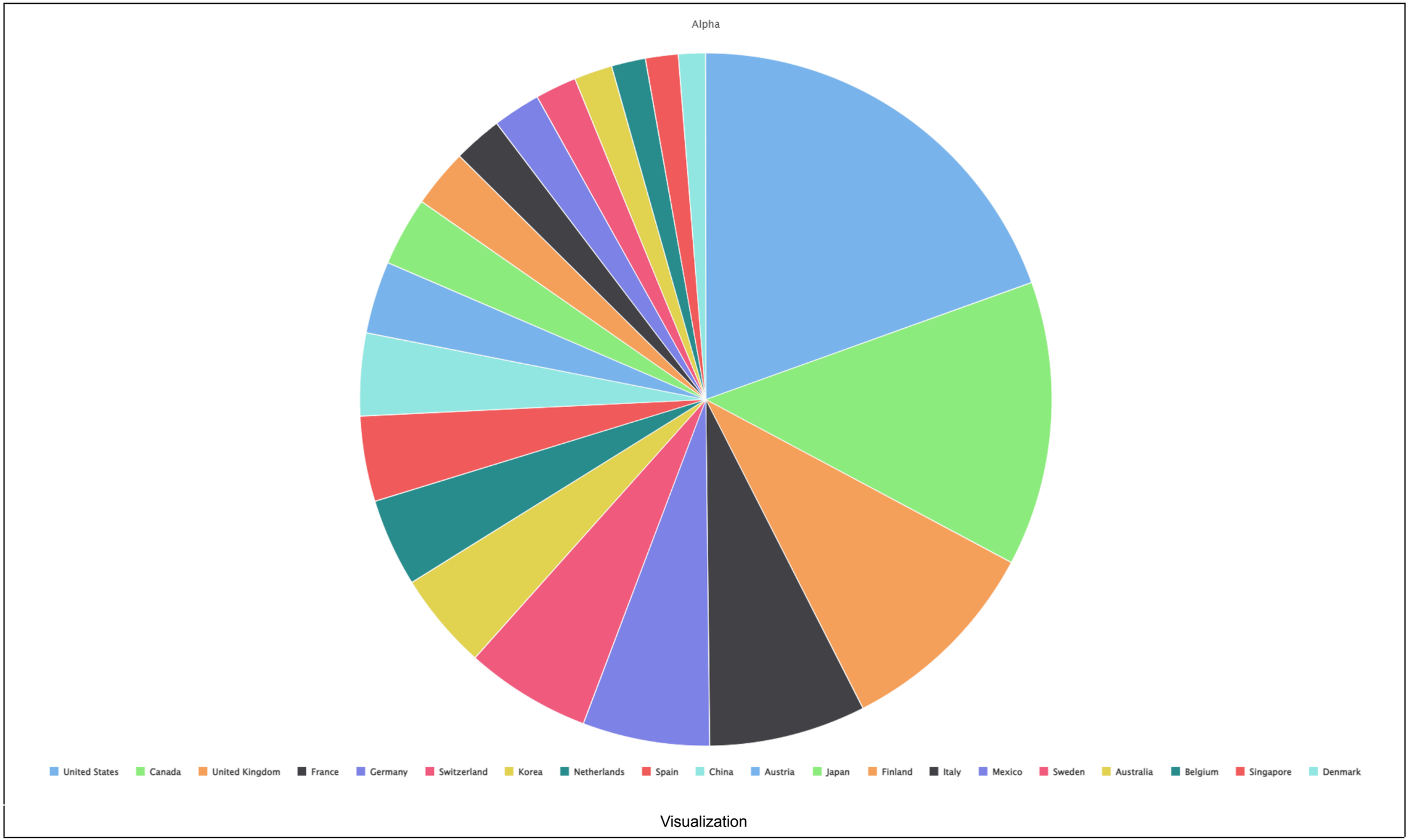
	A	B	C	D	E	F	G	H	I
		country	Values						
		Australia		Austria		Belgium		Brazil	
	productBrand	SUM of grossSa	SUM of profit	SUM of grossSa	SUM of profit	SUM of grossSa	SUM of profit	SUM of grossSa	SUM of pro
	Alpha	655871.83	307639.95	1215390.83	581647.34	582064.06	273348.77		
	Antoni	355996.54	136468.63	1176964.58	438075.76	661591.48	259541.18		
	Blue Steel					928370.93	482555.82	1038934.67	5417(
	BugShield			128695	89579	78129	54819.9	241034	1668(
	Canyon Mule	4458683.78	1956469.6	2435562.11	1036029.49	391167	172007.04	4150620.58	17504
	Course Pro					2090117.12	1288149.74	2067949.33	12689
	Edge								
	Epoch	1560766.52	694719.34	2709830.79	1185258.58	2616575.93	1144369.55	1677634.44	7301(
	EverGlow	36609.3	16426.8	292102.19	136499.4	364027.65	165212.75	615500.07	28
	Extreme	5634265.02	2567229.87	6528829.4	3023199.29	4137569.58	1844776.42		
	Firefly	1195401.09	623187.91	1149895.01	607445.46	657255.4	365411.9	618634.18	269(
	Glacier							330537.44	1292(
	Granite	3830381.93	1721537.77	3535634.75	1583902.52	3222700.42	1460840.27	52218	274
	Hailstorm	1510076.96	783082.01			10295953.25	5211436.32	9249067.68	46738

Spreadsheet Results

	Brand	Country	GrossSales	NetProfit
1	<null>	United Kingdom	1555862.33	<null>
2	<null>	United States	8195967.87	<null>
3	Alpha	Australia	655871.83	307639.95
4	Alpha	Austria	1215390.83	581647.34
5	Alpha	Belgium	582064.06	273348.77
6	Alpha	Canada	5158046.84	2293569.33
7	Alpha	China	1373150.87	666709.11
8	Alpha	Denmark	460986.22	218144.19
9	Alpha	Finland	1027868.76	470705.55
10	Alpha	France	2814171.70	1266222.20
11	Alpha	Germany	2289115.93	1024541.90
12	Alpha	Italy	855931.72	391937.36
13	Alpha	Japan	1224149.90	555296.44

SQL Query Results

For the brand **Alpha:**



Visualization

>> Refine the query to show the gross sales and net profit for each product of the brand

1. Import into separate sheets the fact table and the product dimension.
2. Use VLOOKUP to retrieve the product.productBrand, product.productName, product.unitCost, and product.unitPrice from the dimension sheets. Gross sales are calculated as **=quantity*unitPrice**. Profit is calculated as **=quantity*(unitPrice-unitCost)**. This will yield 6 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: productBrand, productName
    b. Column:
    c. Value: grossSales, profit as *Columns*
5. The pivot table should display the total gross sales and net profit for each product of the brand.

Spreadsheet Instructions

```sql
SELECT
  dp.productBrand,
  dp.productName,
  SUM(fo.quantity * dp.unitPrice) AS gross_sales,
  SUM(fo.quantity * dp.unitPrice) - SUM(fo.quantity * dp.unitCost) AS net_profit
FROM
  fact_orders fo
JOIN
  dim_product dp ON fo.productKey = dp.productKey
GROUP BY
  dp.productName,
  dp.productBrand
ORDER BY
  dp.productBrand,
  dp.productName
```

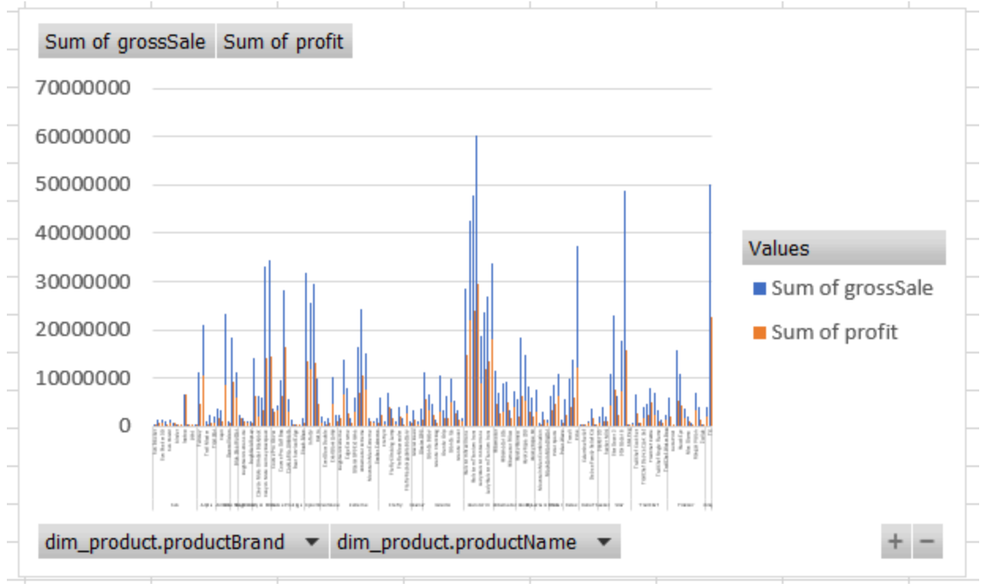
SQL Query

productBrand	productName	SUM of grossSa	SUM of profit
⊟	backpack	819160.38	0
	binder	497705.44	0
	envelopes	366007.36	0
	laptop	6739674.95	0
	notepad	416502.67	0
	pens	546754.56	0
	printer paper	366024.84	0
Total		9751830.2	0
⊟ Alpha	Fairway	11324326.11	4611687.24
	Maximus	20962627.22	10585961.1
	Trail Master	1083685	386266.9
	Trail Scout	2323356	830746.2
	Trail Star	2018016	857394.72
Alpha Total		37712010.33	17272056.16
⊟ Antoni	Bella	3807032.5	1760925.3
	Capri	3469068.36	1209280.49
	Dante	23189072.82	8565022.85
	Opera Vision	1201420	650951.2

Spreadsheet Results

	productBrand	productName	gross_sales	net_profit
1	<null>	backpack	819160.38	<null>
2	<null>	binder	497705.44	<null>
3	<null>	envelopes	366007.36	<null>
4	<null>	laptop	6739674.95	<null>
5	<null>	notepad	416502.67	<null>
6	<null>	pens	546754.56	<null>
7	<null>	printer paper	366024.84	<null>
8	Alpha	Fairway	11324326.11	4611687.24
9	Alpha	Maximus	20962627.22	10585961.10
10	Alpha	Trail Master	1083685.00	386266.90
11	Alpha	Trail Scout	2323356.00	830746.20
12	Alpha	Trail Star	2018016.00	857394.72
13	Antoni	Bella	3807032.50	1760925.30
14	Antoni	Capri	3469068.36	1209280.49
15	Antoni	Dante	23189072.82	8565022.85
16	Antoni	Opera Vision	1201420.00	650951.20

SQL Query Results



Visualization

*>> Refine the query to show the brands in each country that are outperforming the gross sales of brand **Hailstorm.***

1. Import into separate sheets the fact table, the product dimension, and the retailer dimension.
2. Use VLOOKUP to retrieve the product.productBrand, product.unitCost, product.unitPrice, and retailer.country from the dimension sheets. Gross sales are calculated as **=quantity*unitPrice**. Profit is calculated as **=quantity*(unitPrice-unitCost)**. This will yield 6 new columns in the fact sheet.
3. On the fact sheet, create a helper column that will check if the row's product brand has outperformed *Hailstorm* in the same country, i.e., =IF(sum_of_sales_for_this_brand_in_country>sum_of_sales_for_Hailstorm_in_same_country, "Yes", "No"). To reduce computation costs, a combination of INDEX and MATCH functions can retrieve the sum of gross sales from a pivot table setup as follows:
    a. Row: country
    b. Column: productBrand
    c. Value: grossSales
4. Create a pivot table using ALL the columns in the fact sheet.
5. Use the following pivot details:

```
WITH HailstormSales AS (
SELECT
  dr.country AS Country,
  SUM(fo.quantity * dp.unitPrice) AS HailstormGrossSales
FROM
  fact_orders fo
JOIN
  dim_product dp ON fo.productKey = dp.productKey
JOIN
  dim_retailers dr ON fo.retailerKey = dr.retailerKey
WHERE
  dp.productBrand = 'Hailstorm'
GROUP BY
  dr.country
)
SELECT
dp.productBrand AS Brand,
dr.country AS Country,
SUM(fo.quantity * dp.unitPrice) AS GrossSales,
SUM(fo.quantity * (dp.unitPrice - dp.unitCost)) AS NetProfit
FROM
fact_orders fo
JOIN
dim_product dp ON fo.productKey = dp.productKey
JOIN
```

a. Row: productBrand, productName

b. Column:

c. Value: grossSales, profit as *Columns*

d. Filter: helper column (filter by value; untick ‘No’)

6. The pivot table should display the total gross sales for each brand that outperforms *Hailstorm* in the same country.

Spreadsheet Instructions

```
dim_retailers dr ON fo.retailerKey = dr.retailerKey
JOIN
HailstormSales hs ON dr.country = hs.Country
WHERE
dp.productBrand != 'Hailstorm'
GROUP BY
dp.productBrand, dr.country, hs.HailstormGrossSales
HAVING
SUM(fo.quantity * dp.unitPrice) > hs.HailstormGrossSales
ORDER BY
dr.country,
GrossSales DESC;
```

SQL Query

country	productBrand	SUM of grossSa
Australia	Canyon Mule	4458683.78
	Epoch	1560766.52
	Extreme	5634265.02
	Granite	3830381.93
	Husky	2958501.96
Australia Total		18442599.21
Italy	Star	4958854.81
Italy Total		4958854.81
Sweden	Granite	3571373.65
Sweden Total		3571373.65
Grand Total		**26972827.67**

Spreadsheet Results

	Brand	Country	GrossSales
1	Extreme	Australia	5634265.02
2	Canyon Mule	Australia	4458683.78
3	Granite	Australia	3830381.93
4	Husky	Australia	2958501.96
5	Epoch	Australia	1560766.52
6	Star	Italy	4958854.81
7	Granite	Sweden	3571373.65

SQL Query Results

[Not available]

Visualization

Report 2 Comparison and OLAP Analysis

The query filtering data by a specific brand (e.g., "WHERE dp.productBrand = 'Hailstorm'") is an example of a **slice** operation, as it narrows the data down to a single dimension (productBrand). The HAVING SUM(fo.quantity * dp.unitPrice) > hs.HailstormGrossSales is a **dice** operation, which filters based on a condition that compares the sum of gross sales for non-Hailstorm products against Hailstorm's gross sales in each country. The analysis of sales for each product in each product brand is also an example of the OLAP operation **drill-down** since we are going *down* the product hierarchy (Product Line → ProductBrand → **Product**)

Report #3: Performance of the retailers per country per quarter based on gross sales and net profit

1. Import into separate sheets the fact table, the product dimension, and the retailer dimension.
2. Use VLOOKUP to retrieve the product.productBrand, product.unitCost, product.unitPrice and retailer.country from the dimension sheets. Gross sales are calculated as **=quantity*unitPrice**. Profit is calculated as **=quantity*(unitPrice-unitCost)**. Quarter is **=ROUNDUP(MONTH(orderDate)/3,0).** This will yield 7 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: country, retailerKey
    b. Column: quarter
    c. Value: grossSales, profit as *Columns*
5. The pivot table should display the total gross sales and net profit for retailers per country per quarter.

Spreadsheet Instructions

```sql
SELECT
 dr.country AS Country,
 dr.type AS Retailer,
 EXTRACT(YEAR FROM fo.orderDate) AS Year,
 EXTRACT(QUARTER FROM fo.orderDate) AS Quarter,
 SUM(fo.quantity * dp.unitPrice) AS GrossSales,
 SUM(fo.quantity * (dp.unitPrice - dp.unitCost)) AS NetProfit
FROM
 fact_orders fo
JOIN
 dim_retailers dr ON fo.retailerKey = dr.retailerKey
JOIN
 dim_product dp ON fo.productKey = dp.productKey
GROUP BY
 dr.country,
 dr.retailerKey,
 EXTRACT(YEAR FROM fo.orderDate),
 EXTRACT(QUARTER FROM fo.orderDate)
ORDER BY
 dr.country,
 Retailer,
 Year,
 Quarter,
 GrossSales DESC;
```

SQL Query

Row Labels	2015-Q1 Sum of grossSale	Sum of profit	2015-Q2 Sum of grossSale	Sum of profit	2015-Q3 Sum of grossSale	Sum of profit	2015-Q4 Sum of grossSale	Sum of profit	2016-Q Sum of
Australia									**1**
Department Store									
Eyewear Store									
Golf Shop									
Outdoors Shop									
Austria	**1084359.86**	**468767.89**	**1567988.26**	**688123.35**	**1137977.37**	**483190.31**	**1267852.9**	**556949.07**	**1**
Department Store	303985.84	134272.65	686160.11	295625.58	487580.35	210013.79	490892.37	210405.58	
Direct Marketing					7258.64	5119.52	6482.28	4568.16	
Golf Shop	70785.6	29729	115486.83	49537.01	85442.34	34151.5	72054.2	30523.56	
Outdoors Shop	571372.82	246585.59	595189.28	269559.64	405733.7	171304.47	540018.07	245389.31	
Sports Store	138215.6	58180.65	171152.04	73401.12	151962.34	62601.03	158405.98	66062.46	
Belgium	**742035.29**	**344985.32**	**715583.65**	**329618.69**	**768724.27**	**356252.27**	**803940.04**	**375837.66**	**2**
Department Store	112911.67	52809.84	150521.47	68781.16	102819.85	46178.81	85179.27	38122.04	
Eyewear Store	92936.38	40114.71	89506.62	37701.73	84095.67	34560.32	55166.34	23711.72	
Golf Shop	400071.82	198036.69	315283.77	157686.51	361329.94	184049.77	345141.62	177368.19	1
Outdoors Shop					60173.86	26386.75	180663.62	78705.87	
Sports Store	51003.42	22733.06	81598.39	36213.95	106265.97	47537.64	97998.49	43180.21	

Spreadsheet Results

	Country	Retailer	Year	Quarter	GrossSales	NetProfit
1	Australia	Department Store	2016	1	531255.59	234413.21
2	Australia	Department Store	2016	2	505238.41	216163.89
3	Australia	Department Store	2016	3	688567.03	297524.62
4	Australia	Department Store	2016	4	761340.25	325803.77
5	Australia	Department Store	2017	1	689979.49	299044.69
6	Australia	Department Store	2017	2	1481591.92	653932.18
7	Australia	Department Store	2017	3	1193611.39	510105.39
8	Australia	Department Store	2017	4	1090132.01	479664.98
9	Australia	Department Store	2018	1	1020125.94	453319.61
10	Australia	Department Store	2018	2	1557115.23	693721.62
11	Australia	Department Store	2018	3	227354.33	100239.90
12	Australia	Eyewear Store	2016	1	94064.01	49353.26
13	Australia	Eyewear Store	2016	2	79702.74	42063.02
14	Australia	Eyewear Store	2016	3	66202.42	34142.74
15	Australia	Eyewear Store	2016	4	73634.39	37453.39
16	Australia	Eyewear Store	2017	1	151695.20	78263.17
17	Australia	Eyewear Store	2017	2	174803.83	90515.12
18	Australia	Eyewear Store	2017	3	201598.01	103657.36
19	Australia	Eyewear Store	2017	4	155400.26	79560.44

SQL Query Results

[Not available]

Visualization

>> Refine the query to show the retailers' performance based on the quantity sold per product

1. Import into separate sheets the fact table, the product dimension, and the retailer dimension.
2. Use VLOOKUP to retrieve the product.productName and retailer.type from the dimension sheets. This will yield 2 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: productName
    b. Column: retailerType
    c. Value: quantity (shown as % of row total)
5. The pivot table should display the total quantity of products sold per retailer.

Spreadsheet Instructions

```sql
SELECT
  dp.productName AS ProductName,
  dr.type AS RetailerName,
  COALESCE(SUM(fo.quantity) * 100.0 / NULLIF(total_sales.totalQuantity, 0), 0) AS
PercentageOfTotalSales
FROM
  dim_product dp
JOIN
  fact_orders fo ON dp.productKey = fo.productKey
JOIN
  dim_retailers dr ON fo.retailerKey = dr.retailerKey
JOIN (
  SELECT
     retailerKey,
     SUM(quantity) AS totalQuantity
  FROM
     fact_orders
  GROUP BY
     retailerKey
) AS total_sales ON dr.retailerKey = total_sales.retailerKey
GROUP BY
  dp.productName,
  dr.type,
  total_sales.totalQuantity  -- Add total_quantity to the GROUP BY clause
ORDER BY
  dp.productName,
  dr.type;
```

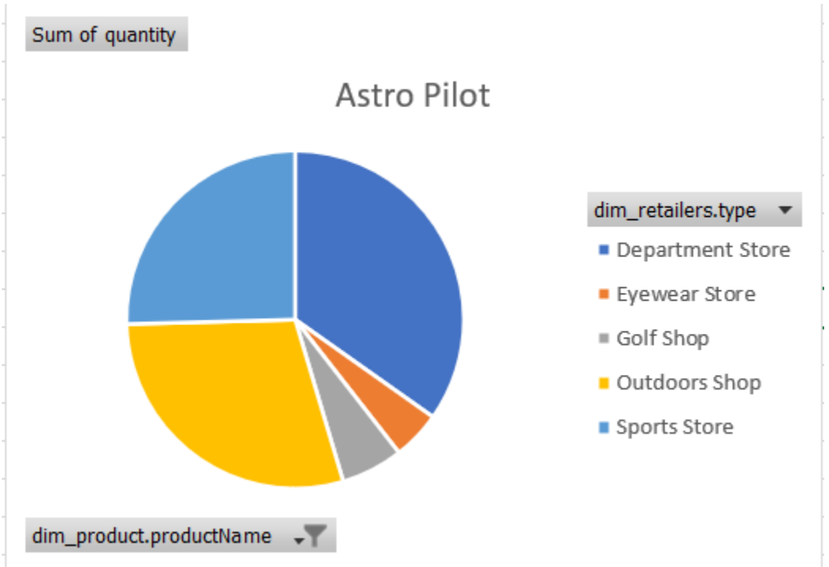
SQL Query

Sum of quantity	Column Labels					
Row Labels	Department Store	Direct Marketing	Eyewear Store	Golf Shop	Outdoors Shop	S
Aloe Relief	73.73%	26.27%	0.00%	0.00%	0.00%	
Astro Pilot	34.78%	0.00%	4.66%	5.95%	29.18%	
Auto Pilot	57.77%	0.00%	0.00%	0.00%	25.51%	
backpack	0.00%	0.00%	0.00%	0.00%	0.00%	
Bear Edge	0.00%	0.00%	0.00%	0.00%	48.86%	
Bear Survival Edge	0.00%	0.00%	0.00%	0.00%	0.00%	
Bella	28.14%	0.00%	2.89%	6.80%	34.95%	
binder	0.00%	0.00%	0.00%	0.00%	0.00%	
Blue Steel Max Putter	25.19%	0.00%	0.00%	62.26%	0.00%	
Blue Steel Putter	54.42%	0.00%	0.00%	43.12%	0.00%	
BugShield Extreme	0.00%	0.00%	0.00%	0.00%	91.33%	
BugShield Lotion	100.00%	0.00%	0.00%	0.00%	0.00%	
BugShield Lotion Lite	74.28%	25.72%	0.00%	0.00%	0.00%	
BugShield Natural	0.00%	0.00%	0.00%	0.00%	10.35%	
BugShield Spray	18.43%	0.00%	0.00%	0.00%	0.00%	
Calamine Relief	0.00%	0.00%	0.00%	0.00%	0.00%	
Canyon Mule Carryall	100.00%	0.00%	0.00%	0.00%	0.00%	
Canyon Mule Climber Backpack	51.86%	0.00%	0.00%	0.00%	26.03%	

Spreadsheet Results

1	Aloe Relief	Department Store	1.12495
2	Aloe Relief	Department Store	0.78083
3	Aloe Relief	Department Store	0.86176
4	Aloe Relief	Department Store	0.46637
5	Aloe Relief	Department Store	0.40543
6	Aloe Relief	Department Store	0.38733
7	Aloe Relief	Department Store	0.32841
8	Aloe Relief	Department Store	0.45527
9	Aloe Relief	Direct Marketing	26.39193
10	Aloe Relief	Direct Marketing	28.29466
11	Aloe Relief	Direct Marketing	29.91925
12	Aloe Relief	Direct Marketing	24.17026
13	Aloe Relief	Direct Marketing	26.22671
14	Aloe Relief	Direct Marketing	15.38902
15	Astro Pilot	Department Store	12.73114
16	Astro Pilot	Department Store	0.50309
17	Astro Pilot	Department Store	0.35868

SQL Query Results



Visualization

>> Refine the query to show which retailers have a total quarterly gross sales of more than 3,000,000 amount

1. Import into separate sheets the fact table, the product dimension, and the retailer dimension.
2. Use VLOOKUP to retrieve the product.unitPrice and retailer.type from the dimension sheets. Gross sales are calculated as **=quantity*unitPrice**. Quarter is **=ROUNDUP(MONTH(orderDate)/3,0)**. This will yield 5 new columns in the fact sheet.
3. On the fact sheet, create a helper column that will check if the row's retailerType has exceeded 3,000,000 total gross sales in that quarter, i.e., =IF(sum_of_gross_sales_for_this_retailer_and_this_quarter>3000000, "Yes", "No"). To reduce computation costs, a combination of INDEX and MATCH functions can retrieve the sum of gross sales from a pivot table setup as follows:
    a. Row: retailerKey
    b. Column: quarter
    c. Value: grossSales
4. Create a pivot table using ALL the columns in the fact sheet.
5. Use the following pivot details:
    a. Row: retailerType, retailerKey
    b. Column: quarter
    c. Value: grossSales
    d. Filter: helper column (filter by value; pick 'Yes')
6.

Spreadsheet Instructions

```sql
SELECT
   dr.retailerKey AS RetailerCode,
   dr.type AS RetailerName,
   COALESCE(SUM(fo.quantity * dp.unitPrice), 0) AS TotalGrossSales
FROM
   dim_product dp
JOIN
   fact_orders fo ON dp.productKey = fo.productKey
JOIN
   dim_retailers dr ON fo.retailerKey = dr.retailerKey
GROUP BY
   dr.retailerKey,
   dr.type,
   dp.productName
HAVING
   TotalGrossSales > 3000000
ORDER BY
   RetailerName,
   dr.retailerKey;
```

SQL Query

Sum of grossSale									
Row Labels	2014-Q4	2015-Q1	2015-Q2	2015-Q3	2015-Q4	2016-Q1	2016-Q2	2016-Q3	2016-Q4
Department Store		29070115.53	29974258.58	28973332.99	31332064.81	33544112.46	27694362.66	31670092.14	3779522
1137		3435518.83	3374081.43	3287236.34	3987564.36	5405654.78	4339259.17	5314075.62	59697
1148		2522880.78	2331795.82	2223948.58	2889703.95	1917179.69	2027430.86	2345937.64	288477
1192		3620035.38	3320797.85	3551024.67	2860643.95	3651209.31	2280441.36	2935817.18	364231
1201		1871542.09	1657641.23	1878726.07	1843879.55	2509100.2	1693781.71	2215052.72	270023
1213		236868.74	542490.37	398116.52	1157894.44	957036.34	1489701.69	1178287.42	183228
1216		885754.7	888105.78	710708.07	910527.17	534365.84	599580.79	486863.95	62933
1218		1736303.11	1974471.21	1518258.4	895340.59	1176671.62	346964.6	891001.63	78651
1223		297513.44	294176.47	270168.54	190107.66	206852.14	288530.81	235154.5	24912
1228		417750.61	578299.57	375148.23	517718.98	296555.7	379750.2	634364.13	70831
1235		775593.06	874403.13	585283.14	896592.23	861685.02	880852.41	1001729.06	112950
1241		992849.91	683382.21	877002.47	653378.31	1110638.65	1133549.76	1056078.47	9872
1250		4993.3	116802.35	266686.85	116252.13	167988.65	242645.12	269900.16	16976
1255						531255.59	505238.41	688567.03	76134
1259		1316373.08	1173362.3	1132335.96	1490400.73	1438669.94	1415237.59	1258081.27	151752
1260		1228611.85	1130869.15	1327368.72	1411864.01	1335522	1555518.02	1826655.8	172256
1272		3912342.44	4118569.17	4172908.41	4612780.28	5449386.94	3395900.34	3930542.77	510988
1282		3720735.18	4237675.63	3936476.24	4056637.67	2651653.69	2507594.68	2812157.23	325793

Spreadsheet Results

	RetailerCode	RetailerName	TotalGrossSales
1	1137	Department Store	3143038.50
2	1137	Department Store	4040763.32
3	1137	Department Store	3527239.52
4	1272	Department Store	4123391.47
5	1272	Department Store	4089844.08
6	1282	Department Store	3096727.14
7	1149	Golf Shop	3104075.79
8	1149	Golf Shop	4137365.40
9	1229	Golf Shop	3645408.78
10	1229	Golf Shop	5216512.32
11	1270	Golf Shop	3925095.04
12	1274	Golf Shop	3468988.08
13	1274	Golf Shop	4359804.40

SQL Query Results

[Not available]

Visualization

Report 3 Comparison and OLAP Analysis

The combination of dr.country, dr.type, and EXTRACT(QUARTER FROM fo.orderDate) in the GROUP BY clause acts as a **dice** operation, filtering the data cube to look at gross sales and net profit for specific combinations of country, retailer type, and quarter. The HAVING TotalGrossSales > 3000000 clause functions as a **slice**, making the query focus only on retailers with total gross sales exceeding 3 million. This eliminates retailers that don't meet the reported quota.

Report #4: Which product(s) did not have a single sale for a given duration? The duration is chosen to be the first quarter of 2018 (January 1 - March 31)

1. Import into separate sheets the fact table and the product dimension.
2. Use VLOOKUP to retrieve the product.productName from the dimension sheets. QuarterYear is **=CONCATENATE(YEAR([@orderDate]),"-Q", ROUNDUP(MONTH([@orderDate])/3, 0))**. This will yield 2 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: productKey
    b. Column:
    c. Value: quantity
    d. Filter: quarterYear (as value; 2018-Q1)
5. If you set the row field to show empty values, and filter the row field as ‘sum of quantity is less than 1’, the pivot table should display the products that did not have a single sale for the selected quarter. Kindly note that the ‘Product Name’ column below was manually added.

Spreadsheet Instructions

```sql
SELECT
   dp.productKey,
   dp.productName
FROM
   dim_product dp
LEFT JOIN fact_orders fo ON dp.productKey = fo.productKey
   AND fo.orderDate BETWEEN '2018-01-01' AND '2018-03-31'
GROUP BY
   dp.productKey,
   dp.productName
HAVING
   SUM(fo.quantity) IS NULL OR SUM(fo.quantity) < 1
ORDER BY
   dp.productName;
```

SQL Query

quarterYear	2018-Q1		
Row Labels	**Sum of quantity**		Product Name
73110			Single Edge
122120			Bella
123120			Capri
123160			Capri
124130			Cat Eye
124140			Cat Eye
125140			Venue
126120			Dante
126130			Dante
127150			Fairwav

Spreadsheet Results

	productKey	productName
1	151120	Astro Pilot
2	151130	Astro Pilot
3	154156	backpack
4	122110	Bella
5	122120	Bella
6	122130	Bella
7	122150	Bella
8	154152	binder
9	123160	Capri
10	123120	Capri
11	123130	Capri

SQL Query Results

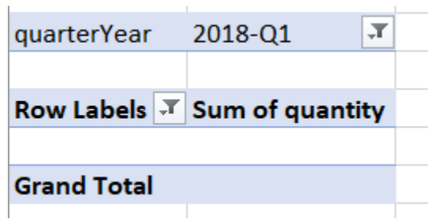
Report #5: Which brand(s) did not have a single sale for a given duration? The duration is chosen to be the first quarter of 2018 (January 1 - March 31)

1. Import into separate sheets the fact table and the product dimension.
2. Use VLOOKUP to retrieve the product.productBrand from the dimension sheets. QuarterYear is **=CONCATENATE(YEAR([@orderDate]),"-Q", ROUNDUP(MONTH([@orderDate])/3, 0))**. This will yield 2 new columns in the fact sheet.
3. Create a pivot table using ALL the columns in the fact sheet.
4. Use the following pivot details:
    a. Row: productBrand
    b. Column:
    c. Value: quantity
    d. Filter: quarterYear (as value; 2018-Q1)
5. If you set the row field to show empty values, and filter row field as: sum of quantity is less than 1, the pivot table should display the brands that did not have a single sale for the selected quarter.

Spreadsheet Instructions

```sql
SELECT
   dp.productBrand
FROM
   dim_product dp
LEFT JOIN fact_orders fo ON dp.productKey = fo.productKey
   AND fo.orderDate BETWEEN '2018-01-01' AND '2018-03-31'
GROUP BY
   dp.productBrand
HAVING
   SUM(fo.quantity) IS NULL OR SUM(fo.quantity) < 1
ORDER BY
   dp.productBrand;
```

SQL Query



quarterYear	2018-Q1
Row Labels	**Sum of quantity**
Grand Total	

Spreadsheet Results

	productBrand
1	<null>

SQL Query Results

Report 4 and 5 Comparison and OLAP Analysis

The OLAP operation focused in these two reports is the **dice** operation since we are focusing on one part of 'time' and restricting the dimensions to a certain period of time (2018 1st quarter specifically). Both approaches used a separate method to narrow down the 'quarter' period, with SQL using the BETWEEN keyword while the Excel method used MONTH() and ROUNDUP() functions.