Subqueries & CTEs

Session Goals:

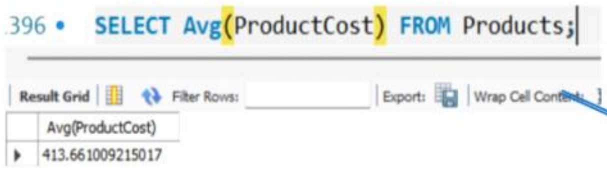
- ✓ Understand Common Table Expressions (CTEs) and why we use them
- ✓ Apply subqueries in SELECT, FROM, WHERE, HAVING, and JOIN
- ✓ Use nested & correlated subqueries for advanced querying
- ✓ Optimize queries using subqueries

Syntax:

```
WITH CTE_NAME AS (
  SELECT column1, column2 FROM table_name WHERE condition
)
SELECT * FROM CTE_NAME;
```

Challenge 1

Find the Products having cost more than the average cost in their subcategory?

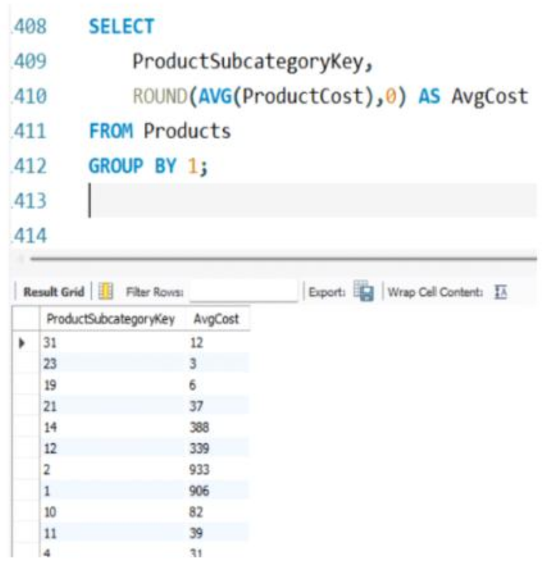


```
SELECT Avg(ProductCost) FROM Products;
```

	Avg(ProductCost)
▶	413.661009215017

Compare every records > avgCost()

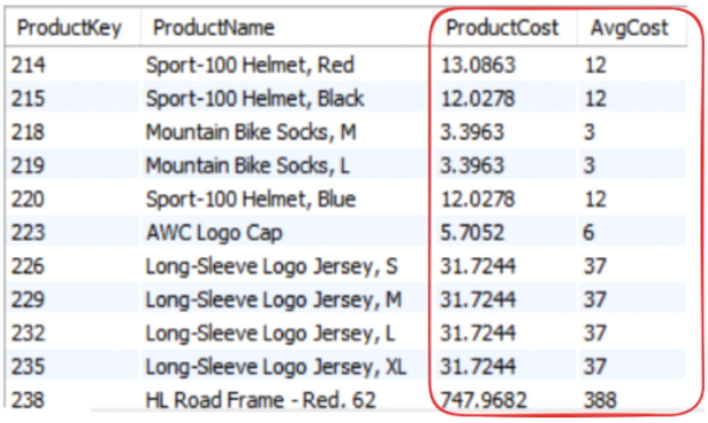
AvgCost with respect to each productSubcategoryKey



```
SELECT
    ProductSubcategoryKey,
    ROUND(AVG(ProductCost),0) AS AvgCost
FROM Products
GROUP BY 1;
```

ProductSubcategoryKey	AvgCost
31	12
23	3
19	6
21	37
14	388
12	339
2	933
1	906
10	82
11	39
4	31

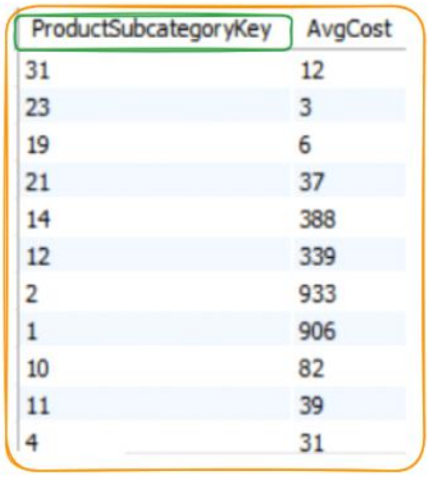
```
-- Find the Products having cost more than the average cost in their subcategory?
WITH AvgCostPerSubcategory AS(
        SELECT
            ProductSubcategoryKey,
            ROUND(AVG(ProductCost),0) AS AvgCost
        FROM Products
    GROUP BY 1
)
SELECT
    p.ProductKey,
    p.ProductName,
    p.ProductCost,
    AvgCost
FROM Products p
JOIN AvgCostPerSubcategory ps
ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey;
```



ProductKey	ProductName	ProductCost	AvgCost
214	Sport-100 Helmet, Red	13.0863	12
215	Sport-100 Helmet, Black	12.0278	12
218	Mountain Bike Socks, M	3.3963	3
219	Mountain Bike Socks, L	3.3963	3
220	Sport-100 Helmet, Blue	12.0278	12
223	AWC Logo Cap	5.7052	6
226	Long-Sleeve Logo Jersey, S	31.7244	37
229	Long-Sleeve Logo Jersey, M	31.7244	37
232	Long-Sleeve Logo Jersey, L	31.7244	37
235	Long-Sleeve Logo Jersey, XL	31.7244	37
238	HL Road Frame - Red, 62	747.9682	388

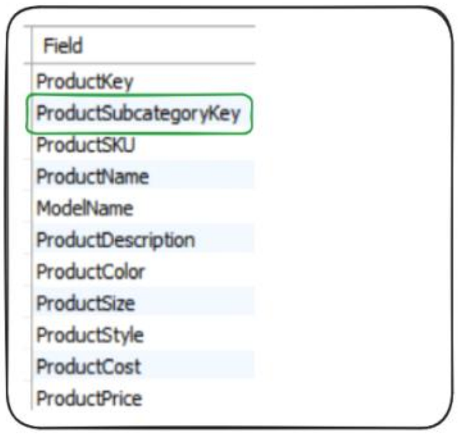
```
-- Find the Products having cost more than the average cost in their subcategory?
WITH AvgCostPerSubcategory AS(
        SELECT
            ProductSubcategoryKey,
            ROUND(AVG(ProductCost),0) AS AvgCost
        FROM Products
    GROUP BY 1
)

SELECT
    p.ProductKey,
    p.ProductName,
    p.ProductCost,
    ps.AvgCost
FROM Products p
JOIN AvgCostPerSubcategory ps
ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
WHERE p.ProductCost > ps.AvgCost
ORDER BY p.ProductCost DESC;
```

Product

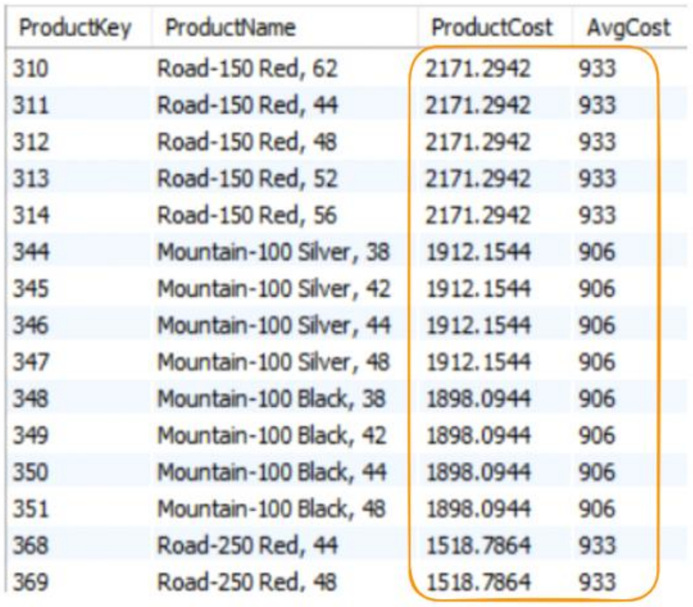


ProductSubcategoryKey	AvgCost
31	12
23	3
19	6
21	37
14	388
12	339
2	933
1	906
10	82
11	39
4	31

Product



Field
ProductKey
ProductSubcategoryKey
ProductSKU
ProductName
ModelName
ProductDescription
ProductColor
ProductSize
ProductStyle
ProductCost
ProductPrice



ProductKey	ProductName	ProductCost	AvgCost
310	Road-150 Red, 62	2171.2942	933
311	Road-150 Red, 44	2171.2942	933
312	Road-150 Red, 48	2171.2942	933
313	Road-150 Red, 52	2171.2942	933
314	Road-150 Red, 56	2171.2942	933
344	Mountain-100 Silver, 38	1912.1544	906
345	Mountain-100 Silver, 42	1912.1544	906
346	Mountain-100 Silver, 44	1912.1544	906
347	Mountain-100 Silver, 48	1912.1544	906
348	Mountain-100 Black, 38	1898.0944	906
349	Mountain-100 Black, 42	1898.0944	906
350	Mountain-100 Black, 44	1898.0944	906
351	Mountain-100 Black, 48	1898.0944	906
368	Road-250 Red, 44	1518.7864	933
369	Road-250 Red, 48	1518.7864	933

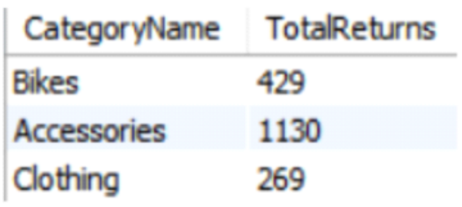
Challenge 2 $\longrightarrow$ Multiple CTE's

Calculate the Total Sales & Total Returns for Each product Category

CategoryReturns

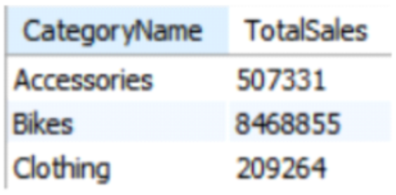
CategorySales

SELECT Statement from the above 2 CTE's



CategoryName	TotalReturns
Bikes	429
Accessories	1130
Clothing	269

```sql
SELECT
    CategoryName,
    SUM(ReturnQuantity) AS TotalReturns
FROM returns r
JOIN Products p
ON r.ProductKey = p.ProductKey
JOIN `product-subcategories` ps
ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
JOIN `product-categories` pc
ON pc.ProductCategoryKey = ps.ProductCategoryKey
GROUP BY 1;
```



CategoryName	TotalSales
Accessories	507331
Bikes	8468855
Clothing	209264

```sql
SELECT
    CategoryName,
    ROUND(SUM(OrderQuantity * ProductPrice),0) AS TotalSales
FROM `Sales-2017` s
JOIN Products p
ON s.ProductKey = p.ProductKey
JOIN `product-subcategories` ps
ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
JOIN `product-categories` pc
ON pc.ProductCategoryKey = ps.ProductCategoryKey
GROUP BY 1;
```

```sql
-- Calculate the Total Sales & Total Returns for Each product Category
WITH CategoryReturns AS(
    SELECT
        CategoryName,
        SUM(ReturnQuantity) AS TotalReturns
    FROM returns r
    JOIN Products p
    ON r.ProductKey = p.ProductKey
    JOIN `product-subcategories` ps
    ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
    JOIN `product-categories` pc
    ON pc.ProductCategoryKey = ps.ProductCategoryKey
    GROUP BY 1
),
CategorySales AS(
    SELECT
        CategoryName,
        ROUND(SUM(OrderQuantity * ProductPrice),0) AS TotalSales
    FROM `Sales-2017` s
    JOIN Products p
    ON s.ProductKey = p.ProductKey
    JOIN `product-subcategories` ps
    ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
    JOIN `product-categories` pc
    ON pc.ProductCategoryKey = ps.ProductCategoryKey
    GROUP BY 1
)
SELECT
    cs.CategoryName,
    cr.TotalReturns,
    cs.TotalSales
FROM CategoryReturns cr
JOIN CategorySales cs
ON cr.CategoryName = cs.CategoryName;
```

```
SELECT
    c.CustomerKey,
    CONCAT(c.FirstName, " ", c.LastName) AS CustomerName,
    SUM(s.OrderQuantity) AS TotalSalesQty
FROM(
    SELECT * FROM `sales-2015`
    UNION ALL
    SELECT * FROM `sales-2016`
    UNION ALL
    SELECT * FROM `sales-2017`
) s
JOIN Customers c
ON s.CustomerKey = c.CustomerKey
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 5;
```

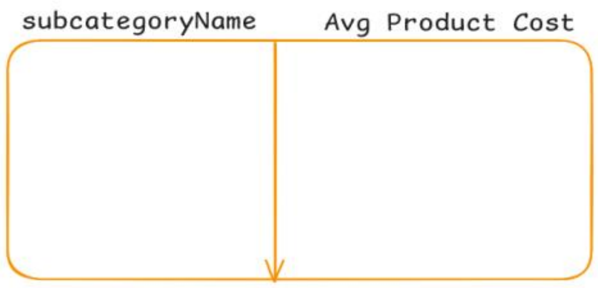
```
WITH AppendSales AS (
    SELECT * FROM `sales-2015`
    UNION ALL
    SELECT * FROM `sales-2016`
    UNION ALL
    SELECT * FROM `sales-2017`
)
SELECT
    c.CustomerKey,
    CONCAT(c.FirstName, " ", c.LastName) AS CustomerName,
    SUM(s.OrderQuantity) AS TotalSalesQty
FROM AppendSales s
JOIN Customers c
ON s.CustomerKey = c.CustomerKey
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 5;
```

Subquery

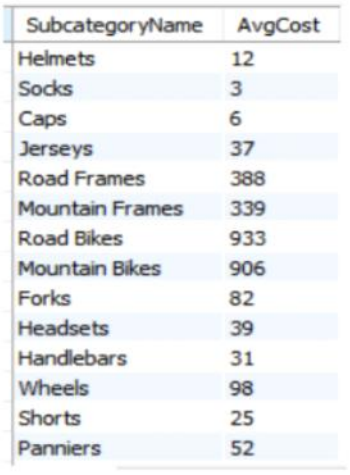
1. SELECT
2. FROM
3. WHERE
4. JOINS
5. HAVING

1. Subquery IN SELECT Clause

-> Find the each subcategoryName with average Product Cost.



```
SELECT
    SubcategoryName,
    ROUND(AVG(ProductCost),0) AS AvgCost
FROM Products p
JOIN `product-subcategories` ps
ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey
GROUP BY 1;
```



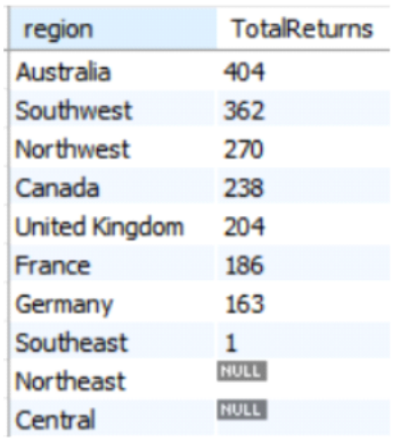
SubcategoryName	AvgCost
Helmets	12
Socks	3
Caps	6
Jerseys	37
Road Frames	388
Mountain Frames	339
Road Bikes	933
Mountain Bikes	906
Forks	82
Headsets	39
Handlebars	31
Wheels	98
Shorts	25
Panniers	52

```
SELECT SubcategoryName, (
    SELECT ROUND(AVG(p.ProductCost),0)
    FROM Products p
    WHERE p.ProductSubcategoryKey = ps.ProductSubcategoryKey
) AS AvgCost
FROM `product-subcategories` ps ;
```

-> Find the each region with total return Qty.

```
SELECT
    region,
    SUM(ReturnQuantity) AS TotalReturns
FROM territories t
LEFT JOIN returns r
ON r.TerritoryKey = t.SalesTerritoryKey
GROUP BY 1
ORDER BY TotalReturns DESC;
```

```
SELECT region, (
    SELECT SUM(ReturnQuantity)
    FROM returns r
    WHERE r.TerritoryKey = t.SalesTerritoryKey
) AS TotalReturns
FROM territories t
ORDER BY TotalReturns DESC;
```



region	TotalReturns
Australia	404
Southwest	362
Northwest	270
Canada	238
United Kingdom	204
France	186
Germany	163
Southeast	1
Northeast	NULL
Central	NULL