CASE Statements in SQL

SYNTAX:

```
SELECT column1,
       CASE
           WHEN condition1 THEN result1
           WHEN condition2 THEN result2
           ELSE default_result
       END AS alias_name
FROM table_name;
```

Conditional Aggregation with CASE:

```
SELECT column,
       SUM(CASE WHEN condition THEN value ELSE 0 END) AS label
FROM table
GROUP BY column;
```

Case Statement - It tries to build a new column based on conditions.
- WHEN THEN

```
mysql> SELECT
    ->  CustomerKey,
    ->     AnnualIncome,
    ->     CASE
    ->          WHEN AnnualIncome < 50000 THEN 'Low Income'
    ->         WHEN AnnualIncome BETWEEN 50000 AND 100000 THEN 'Moderate Income'
    ->         ELSE 'High Income'
    ->     END AS Income_category
    -> FROM Customers
    -> LIMIT 20;
```

CustomerKey	AnnualIncome	Income_category
11000	90000	Moderate Income
11001	60000	Moderate Income
11002	60000	Moderate Income
11003	NULL	High Income
11004	80000	Moderate Income
11005	70000	Moderate Income
11007	60000	Moderate Income
11008	60000	Moderate Income
11009	70000	Moderate Income
11010	70000	Moderate Income
11011	60000	Moderate Income
11012	100000	Moderate Income
11013	100000	Moderate Income
11014	100000	Moderate Income
11015	NULL	High Income
11016	30000	Low Income
11017	20000	Low Income
11018	30000	Low Income
11019	40000	Low Income
11020	40000	Low Income

20 rows in set (0.00 sec)

Misleading??

```
mysql> SELECT
    ->  CustomerKey,
    ->     AnnualIncome,
    ->     CASE
    ->          WHEN AnnualIncome < 50000 THEN 'Low Income'
    ->         WHEN AnnualIncome BETWEEN 50000 AND 100000 THEN 'Moderate Income'
    ->         WHEN AnnualIncome IS NULL THEN 'Not Available'
    ->         ELSE 'High Income'
    ->     END AS Income_category
    -> FROM Customers
    -> LIMIT 20;
```

CustomerKey	AnnualIncome	Income_category
11000	90000	Moderate Income
11001	60000	Moderate Income
11002	60000	Moderate Income
11003	NULL	Not Available
11004	80000	Moderate Income
11005	70000	Moderate Income
11007	60000	Moderate Income
11008	60000	Moderate Income
11009	70000	Moderate Income
11010	70000	Moderate Income
11011	60000	Moderate Income
11012	100000	Moderate Income
11013	100000	Moderate Income
11014	100000	Moderate Income
11015	NULL	Not Available
11016	30000	Low Income
11017	20000	Low Income
11018	30000	Low Income
11019	40000	Low Income
11020	40000	Low Income

20 rows in set (0.00 sec)

Add the Income Category Column in Customer Table with Values using CASE Statement.

```
mysql> ALTER TABLE Customers
    -> ADD COLUMN IncomeCategory Varchar(50)
    -> AFTER AnnualIncome;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> DESC Customers;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| CustomerKey      | int         | NO   | PRI | NULL    |       |
| Prefix           | text        | YES  |     | NULL    |       |
| FirstName        | text        | YES  |     | NULL    |       |
| LastName         | text        | YES  |     | NULL    |       |
| FullName         | varchar(100)| YES  |     | NULL    |       |
| DateOfBirth      | text        | YES  |     | NULL    |       |
| MaritalStatus    | text        | YES  |     | NULL    |       |
| COUNTRY          | varchar(50) | YES  |     | NULL    |       |
| Gender           | text        | YES  |     | NULL    |       |
| EmailAddress     | varchar(100)| YES  |     | NULL    |       |
| AnnualIncome     | int         | YES  |     | NULL    |       |
| IncomeCategory   | varchar(50) | YES  |     | NULL    |       |
| TotalChildren    | int         | YES  |     | NULL    |       |
| EducationLevel   | text        | YES  |     | NULL    |       |
| Occupation       | text        | YES  |     | NULL    |       |
| HomeOwner        | text        | YES  |     | NULL    |       |
| Phone_number     | bigint      | YES  |     | NULL    |       |
| Formatted_Number | varchar(25) | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
18 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT IncomeCategory
    -> FROM Customers;
+----------------+
| IncomeCategory |
+----------------+
| NULL           |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> UPDATE Customers
    -> SET IncomeCategory =
    -> CASE
    ->  WHEN AnnualIncome < 50000 THEN 'Low Income'
    ->  WHEN AnnualIncome BETWEEN 50000 AND 100000 THEN 'Moderate Income'
    ->  WHEN AnnualIncome IS NULL THEN 'Not Available'
    ->  ELSE 'High Income'
    -> END;
Query OK, 2062 rows affected (0.16 sec)
Rows matched: 2062  Changed: 2062  Warnings: 0
```

```
mysql> SELECT DISTINCT IncomeCategory FROM Customers;
+-----------------+
| IncomeCategory  |
+-----------------+
| Moderate Income |
| Not Available   |
| Low Income      |
| High Income     |
+-----------------+
4 rows in set (0.03 sec)
```

```sql
-- Can we add and update income_category in our customer table.
ALTER TABLE Customers
ADD COLUMN IncomeCategory Varchar(50)
AFTER AnnualIncome;

UPDATE Customers
SET IncomeCategory =
CASE
    WHEN AnnualIncome < 50000 THEN 'Low Income'
    WHEN AnnualIncome BETWEEN 50000 AND 100000 THEN 'Moderate Income'
    WHEN AnnualIncome IS NULL THEN 'Not Available'
    ELSE 'High Income'
END;
```

Conditional Aggregation with CASE:

```sql
SELECT column,
       SUM(CASE WHEN condition THEN value ELSE 0 END) AS label
FROM table
GROUP BY column;
```

```
mysql> SELECT
    ->  territoryKey,
    ->     SUM(
    ->          CASE
    ->                  WHEN OrderQuantity > 2 THEN OrderQuantity
    ->              ELSE 0
    ->         END) AS High_sales,
    ->     SUM(
    ->          CASE
    ->                  WHEN OrderQuantity BETWEEN 1 AND 2 THEN OrderQuantity
    ->              ELSE 0
    ->         END) AS Medium_sales,
    ->  SUM(
    ->          CASE
    ->                  WHEN OrderQuantity < 1 THEN OrderQuantity
    ->              ELSE 0
    ->         END) AS Low_sales
    -> FROM `sales-2017`
    -> GROUP BY territoryKey;
```

territoryKey	High_sales	Medium_sales	Low_sales
1	801	5988	0
4	1284	8106	0
6	852	5008	0
10	630	4463	0
8	600	3786	0
9	1185	8302	0
7	498	3738	0
5	0	35	0
3	0	14	0
2	3	21	0

10 rows in set (0.06 sec)

```sql
SELECT
    territoryKey,
    SUM(
        CASE
            WHEN OrderQuantity > 2 THEN OrderQuantity
            ELSE 0
        END) AS High_sales,
    SUM(
        CASE
            WHEN OrderQuantity BETWEEN 1 AND 2 THEN OrderQuantity
            ELSE 0
        END) AS Medium_sales,
    SUM(
        CASE
            WHEN OrderQuantity < 1 THEN OrderQuantity
            ELSE 0
        END) AS Low_sales
FROM `sales-2017`
GROUP BY territoryKey;
```

	OrderQty	HighSales	Moderate_Sales
1	1	0	1
1	0	0	0
1	2	0	2
1	3	3	0
1	4	4	0
1	5	5	0
1	1	0	1
1	0	0	0
1	2	0	2
1	2	0	2
1	2	0	2

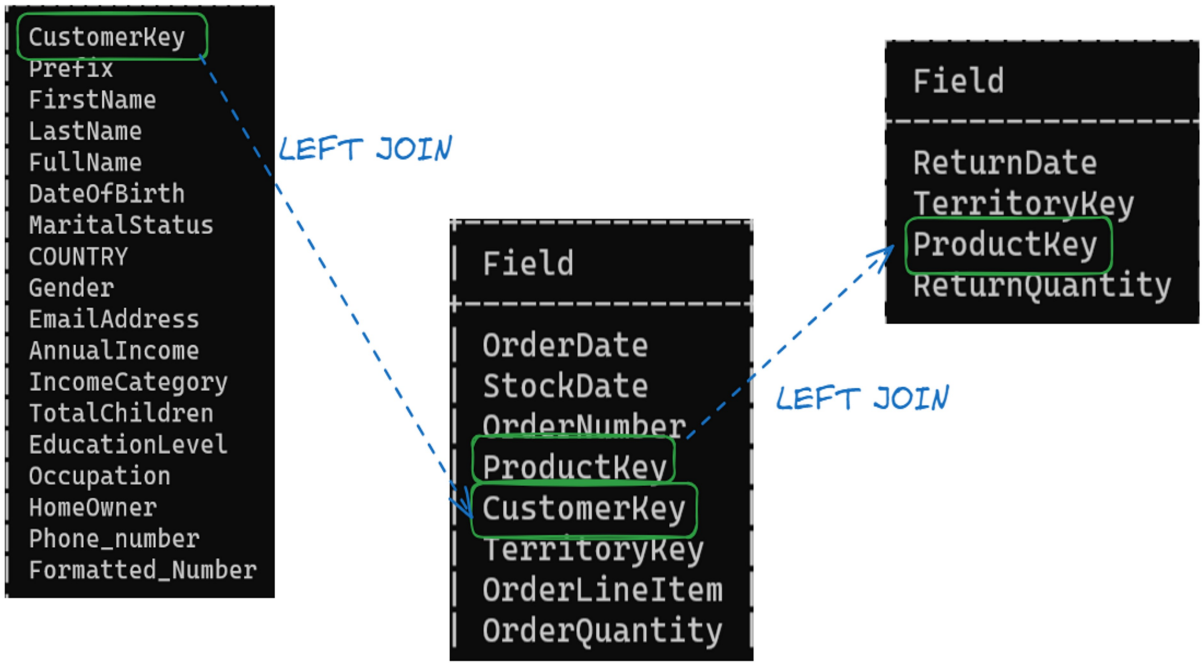
Challenge - 1

Classify customers based on how frequently they return products.

> returnQty [10] - 'Frequent Returner'

returnQty [1 - 10] - 'Occasional Returner'

Else 'Non Returner'

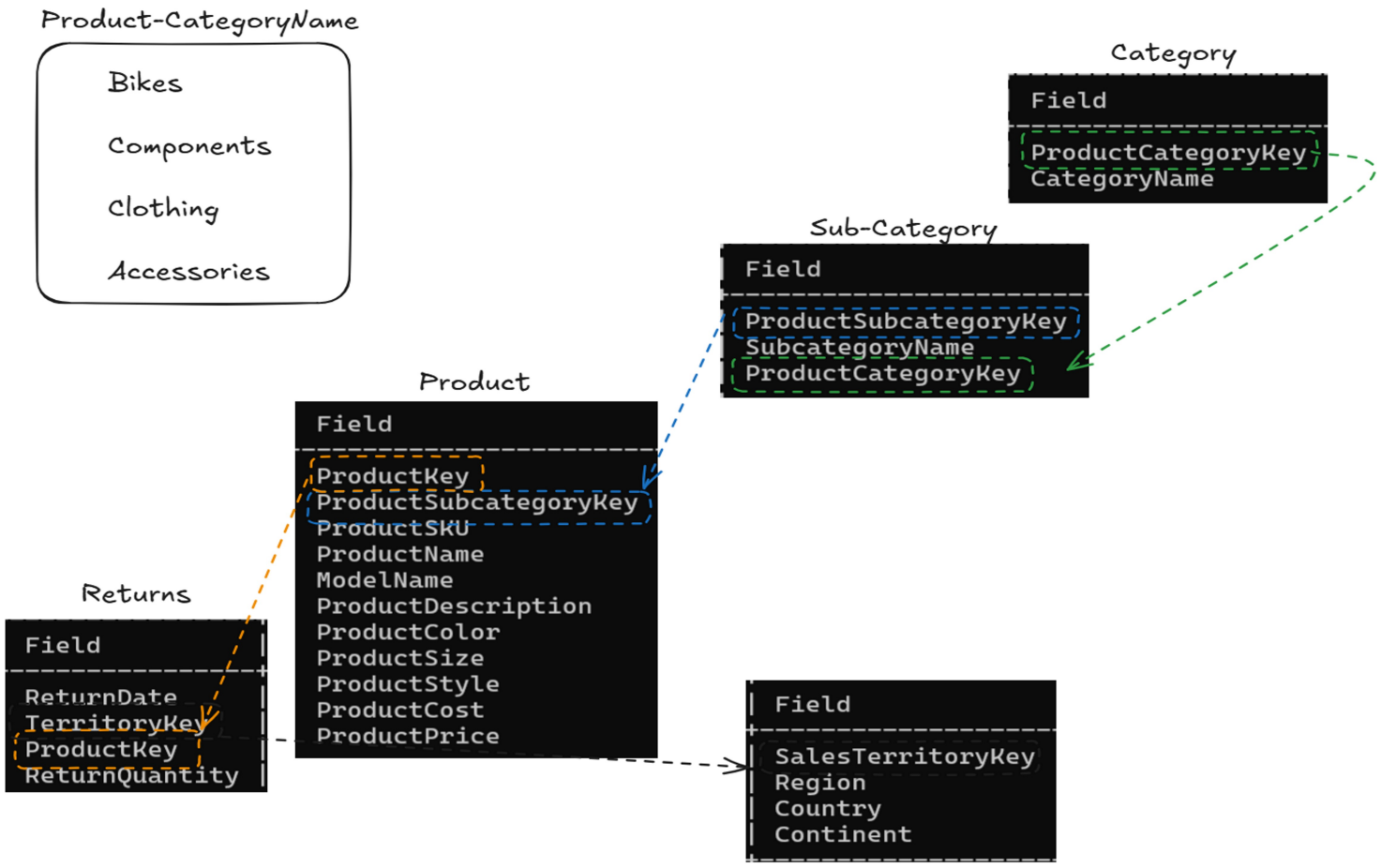



```sql
SELECT
    c.CustomerKey,
    Concat(c.FirstName , ' ', c.LastName) AS CustomerName,
    SUM(COALESCE(r.ReturnQuantity,0)) AS TotalReturns,
    CASE
        WHEN SUM(COALESCE(r.ReturnQuantity,0)) > 10 THEN 'Frequent Returner'
        WHEN SUM(COALESCE(r.ReturnQuantity,0)) BETWEEN 1 AND 10 THEN 'Occasional Returner'
        ELSE 'Non Returner'
    END As ReturnBehaviour
FROM Customers AS c
LEFT JOIN `sales-2015` AS s
ON c.CustomerKey = s.CustomerKey
LEFT JOIN Returns AS r
ON r.ProductKey = s.ProductKey
GROUP BY 1,2
ORDER BY TotalReturns DESC
LIMIT 50;
```

CustomerKey	CustomerName	TotalReturns	ReturnBehaviour
11612	KEITH ANDERSEN	21	Frequent Returner
12286	NATHAN JENKINS	21	Frequent Returner
12226	FAITH WARD	21	Frequent Returner
12585	IAN MARTINEZ	21	Frequent Returner
13076	CRAIG MUNOZ	21	Frequent Returner
12703	RENEE SUAREZ	21	Frequent Returner
13024	MARC MORENO	21	Frequent Returner
13090	WAYNE TANG	21	Frequent Returner
12380	OMAR YUAN	21	Frequent Returner
12575	JACLYN SHEN	21	Frequent Returner
12287	ANDRE ARUN	21	Frequent Returner
12347	SUMMER MADAN	21	Frequent Returner
12991	JÃ?SUS SERRANO	21	Frequent Returner
12989	CARLY GOEL	21	Frequent Returner
12264	BRUCE ASHE	21	Frequent Returner
12335	FRANCISCO ASHE	21	Frequent Returner
12214	JULIE RAJE	21	Frequent Returner
12341	KATIE KUMAR	21	Frequent Returner
12681	CHRISTY CHOW	21	Frequent Returner
11605	HEIDI SUBRAM	21	Frequent Returner

Challenge - 2

Show total return quantities for each product category as columns by region.



```sql
SELECT
    t.region,
    SUM(CASE WHEN pc.CategoryName = "Bikes" THEN r.ReturnQuantity
        ELSE 0 END ) AS BikeReturns,
    SUM(CASE WHEN pc.CategoryName = "Accessories" THEN r.ReturnQuantity
        ELSE 0 END ) AS AccessoriesReturns,
    SUM(CASE WHEN pc.CategoryName = "Clothing" THEN r.ReturnQuantity
        ELSE 0 END ) AS ClothingReturns,
    SUM(CASE WHEN pc.CategoryName = "Components" THEN r.ReturnQuantity
        ELSE 0 END ) AS ComponentReturns
FROM Returns AS r
JOIN Products AS p
ON p.ProductKey = r.ProductKey
JOIN `product-subcategories` AS ps
ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
JOIN `product-categories` AS pc
ON pc.ProductCategoryKey = ps.ProductCategoryKey
JOIN territories t
ON t.SalesTerritoryKey = r.TerritoryKey
GROUP BY t.region;
```

Order of execution:
FROM
JOIN
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT

region	BikeReturns	AccessoriesReturns	ClothingReturns	ComponentReturns
Australia	125	223	56	0
United Kingdom	51	129	24	0
Germany	53	88	22	0
Southwest	78	231	53	0
Canada	22	165	51	0
Northwest	58	172	40	0
France	42	121	23	0
Southeast	0	1	0	0

Home Work - 1

Group By

Calculate total revenue per category-region pair and tag as Low, Moderate, or High Revenue.

SUM ProductPrice * OrderQuantity

Home Work - 2

- Challeng2 + HomeWork1

Classify the Product By its Return Volume

Identify products with High, Moderate, or Low returns across regions and categories.