Filtering in SQL DPP

Sales Transaction Dataset

1. Basic Filtering with WHERE

Write a query to find all transactions where the amount is greater than $1000.

```
SELECT *
FROM sales_transactions
WHERE amount > 1000;
```

Explanation: This query uses the WHERE clause to filter rows and return only those transactions whose amount exceeds $1000, helping identify high-value transactions.

2. Using Logical Operators (AND, OR)

Find all transactions in the Electronics category where the amount is more than $500.

```
SELECT *
FROM sales_transactions
WHERE category = 'Electronics'
  AND amount > 500;
```

Explanation: The AND operator is used to apply multiple conditions simultaneously, ensuring that only Electronics transactions with an amount greater than $500 are selected.

3. Filtering with Date Conditions

Retrieve all transactions that occurred after March 1, 2024.

```
SELECT *
FROM sales_transactions
WHERE transaction_date > '2024-03-01';
```

Explanation: This query filters records based on a date condition to analyze recent transactions occurring after a specific date.

4. Handling Multiple Conditions

Find transactions where the amount is between $500 and $1000 AND the category is Furniture.

```
SELECT *
FROM sales_transactions
WHERE amount BETWEEN 500 AND 1000
  AND category = 'Furniture';
```

Explanation: BETWEEN is used to define a numeric range, while AND ensures that only Furniture category transactions within the specified amount range are returned.

5. Using NULL Filtering

If some transactions have missing payment methods, find those transactions.

```
SELECT *
FROM sales_transactions
WHERE payment_method IS NULL;
```

Explanation: IS NULL is used to identify records with missing or undefined values, which is important for data quality checks.

6. Sorting Results with ORDER BY

Retrieve all transactions sorted by amount in descending order.

```
SELECT *
FROM sales_transactions
ORDER BY amount DESC;
```

Explanation: ORDER BY with DESC sorts the results from highest to lowest amount, making it easier to analyze top transactions first.

7. Counting Transactions per Category (GROUP BY)

Find the number of transactions in each category.

```
SELECT category, COUNT(*) AS transaction_count
FROM sales_transactions
GROUP BY category;
```

Explanation: GROUP BY aggregates data by category, and COUNT is used to calculate how many transactions exist in each group.

8. Using HAVING to Filter Aggregated Data

Retrieve categories that have more than 3 transactions.

```
SELECT category, COUNT(*) AS transaction_count
FROM sales_transactions
GROUP BY category
HAVING COUNT(*) > 3;
```

Explanation: HAVING is used to filter aggregated results, allowing conditions to be applied after grouping the data.

9. Finding the Total Revenue per Region

Calculate the total sales amount per region, displaying only regions where total sales exceed $3000.

```
SELECT region, SUM(amount) AS total_sales
FROM sales_transactions
GROUP BY region
HAVING SUM(amount) > 3000;
```

Explanation: SUM calculates total revenue per region, while HAVING filters regions based on the aggregated sales value.

10. Advanced Query: Filtering High-Value Transactions with Aggregates

Find the regions where the average transaction amount is greater than $800, but only for categories that have more than 3 transactions.

```
SELECT region
FROM sales_transactions
GROUP BY region, category
HAVING AVG(amount) > 800
   AND COUNT(*) > 3;
```

Explanation: This query combines AVG and COUNT with HAVING to identify regions with consistently high-value transactions while ensuring sufficient transaction volume.