**Using split() and explode():**

Extract each individual tag from the tags column and create a new row for each tag, keeping the order\_id.

Count how many orders belong to each unique tag.

**Using regexp\_extract():**

From the details column, extract the serial number (e.g., "A1B2C3D4") for products where the detail starts with "Serial: ".

From the features column, extract the numeric ID (e.g., "123") that appears after "user\_".

**Using translate() and substring():**

Create a new column by removing all vowels (both uppercase and lowercase) from the details column.

Extract the first 5 characters of the features column.

**Using concat\_ws():**

Create a new column that combines the product and quantity columns into a single string like "Laptop (1)".

Combine the elements of the related\_products string into a list using a comma as a separator (you might need split() first).

**Using MapType, explode(), map\_keys(), and map\_values():**

Explode the product\_attributes map to have one row per attribute key-value pair, keeping the order\_id and product.

Get a list of all unique attribute keys present in the product\_attributes column across all orders.

Extract the value associated with the key "color" from the product\_attributes map.

**Using collect\_list() and collect\_set():**

For each customer\_id, create a list of all products they have ordered (collect\_list).

For each customer\_id, create a set of unique products they have ordered (collect\_set).

**Using sample() and sampleBy():**

Take a random sample of 20% of the rows from the DataFrame.

Perform a stratified sample on the product column, taking 50% of "Laptop" orders and 100% of "Mouse" orders.

**Combining split(), explode(), and Aggregation:**

Find the total quantity of products ordered for each unique tag.

**Using substring() and concat\_ws():**

Create a new column that takes the first 3 characters of the product name and concatenates it with the order\_id, separated by a hyphen (e.g., "Lap-1").

**Using regexp\_replace() and translate():**

Remove all non-digit characters from the details column.

Replace all occurrences of the letter 'e' (case-insensitive) in the product column with the character '@'.

**Working with MapType and Filtering:**

Filter the DataFrame to show only the orders where the product\_attributes map contains the key "color" and its value is "silver".

**Using collect\_list() with struct():**

For each order\_date, collect a list of structs containing the product and quantity for all orders placed on that date.

**Advanced pivot():**

Pivot the data to show the total price and total quantity for each product across different customer\_ids.

**Combining sample() and groupBy():**

Take a random sample of 50% of the data and then group the sampled data by customer\_id to find the total number of orders for each customer in the sample.