To implement MapReduce functions for the given requirements, we will assume the input data is structured in a collection of key-value pairs (Tuples), where each tuple represents a book with fields: Title, Author\_name, and Borrowed\_status, and price.

Below are the steps for implementing the required MapReduce functions for each of the 3 tasks.

### Input Format:

Each record represents a book in the following structure:

```

(title, author\_name, borrowed\_status, price)

```

Where:

- `title`: The name of the book (string).

- `author\_name`: The name of the author (string).

- `borrowed\_status`: Whether the book is borrowed or not (boolean, "True" or "False").

- `price`: The price of the book (integer).

### 1. \*\*Display Author-wise list of books\*\*

In this task, we need to list all books by each author, regardless of their borrowed status or price.

#### Map Function:

- The mapper emits the author as the key and the title of the book as the value.

#### Reduce Function:

- The reducer aggregates all the book titles associated with each author and returns the author with a list of their books.

\*\*Map Function:\*\*

```python

def map\_function(book):

title, author\_name, borrowed\_status, price = book

# Emit the author\_name as the key and title as the value

emit(author\_name, title)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(author\_name, titles):

# Aggregate the titles of all books written by the author

return (author\_name, list(titles))

```

### 2. \*\*Display Author-wise list of books having Borrowed status as "True"\*\*

In this task, we need to display a list of books by each author that are borrowed.

#### Map Function:

- The mapper emits the author as the key and the title of the book if the borrowed\_status is "True".

#### Reduce Function:

- The reducer aggregates all the book titles associated with each author and returns the author with a list of their borrowed books.

\*\*Map Function:\*\*

```python

def map\_function(book):

title, author\_name, borrowed\_status, price = book

# Emit the author\_name as the key and title if borrowed\_status is "True"

if borrowed\_status == "True":

emit(author\_name, title)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(author\_name, titles):

# Aggregate the titles of all borrowed books written by the author

return (author\_name, list(titles))

```

### 3. \*\*Display Author-wise list of books having price greater than 300\*\*

In this task, we need to display a list of books by each author whose price is greater than 300.

#### Map Function:

- The mapper emits the author as the key and the title of the book if the price is greater than 300.

#### Reduce Function:

- The reducer aggregates all the book titles associated with each author and returns the author with a list of their expensive books.

\*\*Map Function:\*\*

```python

def map\_function(book):

title, author\_name, borrowed\_status, price = book

# Emit the author\_name as the key and title if price is greater than 300

if price > 300:

emit(author\_name, title)

```

\*\*Reduce Function:\*\*

```python

def reduce\_function(author\_name, titles):

# Aggregate the titles of all books priced greater than 300 written by the author

return (author\_name, list(titles))

```

### Explanation of the Map and Reduce Functions:

1. \*\*Map Function\*\*:

- In the `map\_function`, for each input record (book), we extract the necessary fields. For example, in Task 1, we emit the author name as the key and the book title as the value.

- In Task 2, we include a check to emit only the books with the borrowed status as "True".

- In Task 3, we include a check to emit only the books with a price greater than 300.

2. \*\*Reduce Function\*\*:

- In the `reduce\_function`, for each author, we aggregate the book titles into a list. The `reduce\_function` is responsible for returning the final output for each author.

### Example:

#### Input:

```python

[

("Book 1", "Author A", "True", 400),

("Book 2", "Author B", "False", 250),

("Book 3", "Author A", "True", 150),

("Book 4", "Author B", "True", 350),

("Book 5", "Author A", "False", 500),

("Book 6", "Author C", "True", 350)

]

```

#### Task 1 Output (Author-wise list of all books):

```python

[

("Author A", ["Book 1", "Book 3", "Book 5"]),

("Author B", ["Book 2", "Book 4"]),

("Author C", ["Book 6"])

]

```

#### Task 2 Output (Author-wise list of books with Borrowed status "True"):

```python

[

("Author A", ["Book 1", "Book 3"]),

("Author B", ["Book 4"]),

("Author C", ["Book 6"])

]

```

#### Task 3 Output (Author-wise list of books with price > 300):

```python

[

("Author A", ["Book 1", "Book 5"]),

("Author B", ["Book 4"]),

("Author C", ["Book 6"])

]

```

### Conclusion:

- The MapReduce framework allows for efficient processing of large datasets by using parallel processing. The map function emits key-value pairs, and the reduce function aggregates results based on the key. Each task follows a similar pattern, with the map function filtering data based on specific criteria (borrowed status or price), and the reduce function consolidates the results per author.