### 1. Revisione dello Stato Attuale dei Filtri:

* **Widget di Input Attuali:**
  + TextField singolo (controllato da \_searchController) per la ricerca combinata.
  + (Se hai già implementato filtri separati per Titolo/Autore):
    - TextField per “Titolo” (controllato da \_cercaTitoloController).
    - TextField per “Autore” (controllato da \_cercaAutoreController).
* **Variabili di Stato per le Query Attuali:**
  + \_queryTitolo (String): Valore del filtro per il titolo.
  + \_queryAutore (String): Valore del filtro per l’autore.
* **Logica di Filtro Attuale (\_filterData()):**
  + Recupera i valori da \_cercaTitoloController e \_cercaAutoreController.
  + Filtra \_csvData confrontando i valori estratti dalle righe (usando \_getValueFromRow o indici diretti) con \_queryTitolo e \_queryAutore.
  + La condizione di match è tipicamente valoreRiga.contains(query) (case-insensitive).
  + I filtri sono combinati con logica AND (una riga deve soddisfare tutti i criteri attivi).

### 2. Identificazione e Pianificazione Nuovi Filtri:

* **Campi Candidati per Nuovi Filtri (dal CSV):**
  + Strumento (da keyStrumento): Priorità alta, molto utile per l’utente.
  + Provenienza (da keyArchivioProvenienza): Utile se gli archivi sono distintivi.
  + Volume (da keyVolume): Se i volumi sono significativi per la ricerca.
  + Tipo Multi / Materiale (da keyTipoMulti): Per filtrare per tipo di materiale (es. “Spartito”, “Audio”).
  + Tipo Docu / Documento (da keyTipoDocu): Per filtrare per formato del documento (es. “PDF”, “MP3”).
* **Scelta dei Widget di Input per i Nuovi Filtri:**
  + **Per Strumento:**
    - **Opzione 1 (Consigliata):** DropdownButtonFormField se gli strumenti sono un insieme relativamente limitato e conosciuto.
      * Richiede di popolare le opzioni (staticamente o dinamicamente da \_csvData).
    - **Opzione 2:** TextField se si preferisce ricerca testuale libera.
  + **Per Provenienza:**
    - Simile a Strumento: DropdownButtonFormField preferibile se le opzioni sono limitate.
  + **Per Volume:**
    - TextField è probabilmente adeguato.
  + **Per Tipo Multi / Tipo Docu:** \*DropdownButtonFormField è ideale dato il numero probabilmente limitato di tipi.

### 3. Modifiche all’Interfaccia Utente (UI) dei Filtri:

* **Posizionamento:**
  + Valutare se la bottom dell’AppBar ha spazio sufficiente.
  + **Alternativa Consigliata (se molti filtri):** Un pannello filtri dedicato (es. attivato da IconButton nell’AppBar) implementato come:
    - BottomSheet modale.
    - EndDrawer.
    - AlertDialog (meno ideale per molti input complessi).
* **Widget da Aggiungere (esempi):**
  + Nuovi TextField o DropdownButtonFormField per ogni nuovo criterio di filtro.
  + Labels appropriate e InputDecoration.
  + Pulsante “Applica Filtri” (se non si usa l’aggiornamento automatico) e “Resetta Filtri” nel pannello dedicato.

### 4. Aggiornamenti alla Classe \_CsvViewerScreenState:

* **Nuovi Controller (per TextField):**
  + Es: late TextEditingController \_cercaStrumentoController;
  + Ricordare initState() e dispose().
* **Nuove Variabili di Stato per i Valori dei Filtri:**
  + Es: String \_queryStrumento = ''; (per TextField).
  + Es: String? \_selectedStrumento; (per DropdownButtonFormField, dove null può significare “nessun filtro” o “tutti”).
* **Popolamento Opzioni per DropdownButtonFormField (se usati):**
  + Metodo per estrarre valori unici da \_csvData per un dato campo (es. tutti gli strumenti unici).

**Blocco 6: Sezione 4 (il blocco di codice e la riga successiva)**

dart // Esempio di logica da inserire in \_CsvViewerScreenState List \_getUniqueValuesForColumn(String columnKey) { if (\_csvData.isEmpty) return []; final Set uniqueValues = {}; for (var row in \_csvData) { final value = \_getValueFromRow(row, columnKey); if (value.isNotEmpty) { uniqueValues.add(value); } } final sortedList = uniqueValues.toList()..sort(); return [‘Tutti’]..addAll(sortedList); // Aggiunge un’opzione per non filtrare } —

**Blocco 7: Sezione 5 (fino al secondo blocco di codice)**  
\* Combinare **tutte** le condizioni (matchesTitolo && matchesAutore && matchesStrumento && ...) con &&. \* **Chiamata a setState(() {}):** Mantenuta per aggiornare l’UI. \* ### 6. Gestione Attivazione e Reset dei Filtri:

* **Attivazione:**
  + **Bottone “Applica Filtri”:** Se si usa un pannello dedicato, questo bottone chiamerà \_filterData() e chiuderà il pannello.
  + **Aggiornamento Automatico:** Se i filtri sono direttamente nell’AppBar, considerare l’uso di onChanged per i widget di input (con debounce per i TextField).
* **Pulsante “Resetta Filtri”:**
  + **Logica onPressed:**
    - \_cercaTitoloController.clear();
    - \_cercaAutoreController.clear();
    - \_cercaStrumentoController.clear(); // e altri controller
    - \_queryTitolo = '';
    - \_queryAutore = '';
    - \_queryStrumento = ''; // e altre variabili di query
    - \_selectedStrumento = null; // o 'Tutti', per i dropdown
    - \_filterData();
    - setState(() {}); // Per aggiornare l'UI dei campi di filtro stessi, se necessario

### 7. Flusso Logico Modificato (con Pannello Filtri):

1. Apertura Pannello Filtri: L’utente tocca l’icona “Filtri”.
2. Interazione con Filtri: L’utente imposta i valori nei vari TextField / DropdownButtonFormField nel pannello.
3. Applicazione Filtri: L’utente tocca “Applica Filtri” nel pannello.
   * \_filterData() viene eseguita con i nuovi valori.
   * Il pannello si chiude.
   * La lista \_filteredCsvData si aggiorna.
4. Reset Filtri: L’utente tocca “Resetta Filtri” nel pannello.
   * Tutti i campi di input e le variabili di query vengono resettati.
   * \_filterData() viene eseguita (mostrando tutti i dati).
   * I campi nel pannello si aggiornano per riflettere lo stato resettato.

### 8. Punti Chiave e Considerazioni Aggiuntive:

* **Performance:** Con molti dati e filtri complessi, la performance di \_filterData() potrebbe diventare un problema. Il debouncing è il primo passo; per set di dati enormi, potrebbero essere necessarie ottimizzazioni più avanzate (ma probabilmente non per questo caso d’uso).
* **UX (User Experience):**
  + Fornire un feedback chiaro quando i filtri sono attivi.
  + Assicurare che il reset dei filtri sia intuitivo.
  + Per i DropdownButtonFormField, considerare se l’opzione “Tutti” (o simile) è necessaria o se null è sufficiente per indicare “nessun filtro”.
* **Manutenibilità del Codice:**
  + Mantenere \_filterData() leggibile. Se diventa troppo lunga, considerare di suddividere la logica di match per ogni filtro in piccole funzioni helper.
  + Assicurare coerenza nell’accesso ai dati delle righe (idealmente sempre tramite \_getValueFromRow).