HW1b — Reference Solutions

Questions Covered:

- Relational Algebra 2.1–2.17;
- Tuple Relational Calculus 2.2 and 2.6–2.8.

Part A — Relational Algebra

2.1 Books reviewed by Kim Kardashian

Select the named borrower, join the reviews by card number, and retrieve the corresponding book titles.

```
π_{BOOK.Title, BOOK_REVIEWS.Rating} (
  (σ_{BORROWER.Name = 'Kim Kardashian'}(BORROWER))
  ⋈_{BORROWER.CardNo = BOOK_REVIEWS.CardNo} BOOK_REVIEWS
  ⋈_{BOOK_REVIEWS.BookID = BOOK.BookID} BOOK
)
```

2.2 Staff working in Kim Kardashian's city

Find her city and match that city to the branch location, then retrieve the staff members.

```
π_{LIBRARY_STAFF.Name, LIBRARY_STAFF.Position} (
  (σ_{BORROWER.Name = 'Kim Kardashian'}(BORROWER))
  ⋈_{BORROWER.City = LIBRARY_BRANCH.City} LIBRARY_BRANCH
  ⋈_{LIBRARY_BRANCH.BranchID = LIBRARY_STAFF.BranchID} LIBRARY_STAFF
)
```

2.3 Wishlisted book neither borrowed nor reviewed

Subtract matching (BookID, CardNo) pairs from the wishlist before retrieving borrower information.

```
EligibleWishlist = π_{BookID, CardNo}(NEW_BOOK_WISHLIST)
  − π_{BookID, CardNo}(BOOK_LOANS)
  − π_{BookID, CardNo}(BOOK_REVIEWS)
Answer = π_{Name, CardNo}(BORROWER ⋈ EligibleWishlist)
```

note: Both BookID and CardNo must be part of the comparison.

2.4 macOS reservations by borrowers with book loans

Match macOS computers to reservations, then require that the borrower has at least one qualifying loan.

```
BorrowersWithLoans = π_{CardNo}(BOOK_LOANS)
MacOSComputers = σ_{OperatingSystem = 'macOS'}(LIBRARY_COMPUTER)
BranchNames = π_{BranchID, BranchName}(LIBRARY_BRANCH)
Answer = π_{BORROWER.Name, COMPUTER_RESERVATION.ComputerID,
           BranchName} (
  BORROWER ⋈ BorrowersWithLoans ⋈ COMPUTER_RESERVATION
```

```
    ⋈ MacOSComputers ⋈ BranchNames
)
```

note: "currently" with respect to time is not defined by the schema; a borrower is treated as having a book checked out if they have at least one tuple in BOOK_LOANS. Interpretations based on DateDue are also acceptable.

2.5 Review submitted but corresponding book never borrowed

Compare borrower–book pairs, not borrowers alone.

```
ReviewsWithoutLoans = π_{BookID, CardNo}(BOOK_REVIEWS)
  – π_{BookID, CardNo}(BOOK_LOANS)
Answer = π_{CardNo, Name}(BORROWER ⋈ ReviewsWithoutLoans)
```

2.6 Staff at branches where Steve Bannon borrowed

Find the branch IDs on Steve Bannon's loans, then return the staff at those branches.

```
RelevantBranches = π_{BOOK_LOANS.BranchID} (
  σ_{BORROWER.Name = 'Steve Bannon'} (
    BORROWER ⋈_{BORROWER.CardNo = BOOK_LOANS.CardNo} BOOK_LOANS
  )
)
Answer = π_{Name, Position, Phone}(LIBRARY_STAFF ⋈ RelevantBranches)
```

note: Loans can be from any date.

2.7 Individual rooms at both branches on the same day

Make two copies of the individual-room reservations, one for each branch, and match borrower and date.

```
IndividualReservations = σ_{RoomType = 'Individual'} (
  ROOM_RESERVATION ⋈ MEETING_ROOM ⋈ LIBRARY_BRANCH
)
CharlesVillageReservations = ρ_{CharlesVillageReservations} (
  σ_{BranchName = 'Charles Village'}(IndividualReservations))
TowsonSouthReservations = ρ_{TowsonSouthReservations} (
  σ_{BranchName = 'Towson South'}(IndividualReservations))
MatchingCardNumbers = ρ_{CardNo ← CharlesVillageReservations.CardNo} (
  π_{CharlesVillageReservations.CardNo} (
    CharlesVillageReservations ⋈_{
      CharlesVillageReservations.CardNo = TowsonSouthReservations.CardNo
      AND CharlesVillageReservations.ReservationDate =
          TowsonSouthReservations.ReservationDate}
    TowsonSouthReservations
  )
)
Answer = π_{BORROWER.Name}(BORROWER ⋈ MatchingCardNumbers)
```

note: The requested borrower phone number is unavailable in the schema. Both rooms must be Individual, at different specified branches, on the same date.

2.8 Borrowers who reviewed multiple distinct books

Use a self-product of BOOK_REVIEWS, matching card numbers but requiring different book IDs.

```
FirstReview = ρ_{FirstReview}(BOOK_REVIEWS)
SecondReview = ρ_{SecondReview}(BOOK_REVIEWS)
MultipleReviewCardNumbers = ρ_{CardNo ← FirstReview.CardNo} (
  π_{FirstReview.CardNo} (
    σ_{FirstReview.CardNo = SecondReview.CardNo AND
      FirstReview.BookID ≠ SecondReview.BookID} (
      FirstReview × SecondReview
    )
  )
)
Answer = π_{Name}(BORROWER ⋈ MultipleReviewCardNumbers)
```

2.9 Wishlisted a book also borrowed

Intersect borrower–book pairs in the wishlist and loans.

```
WishlistedAndBorrowed = π_{BookID, CardNo}(NEW_BOOK_WISHLIST)
  ∩ π_{BookID, CardNo}(BOOK_LOANS)
Answer = π_{Name}(BORROWER ⋈ WishlistedAndBorrowed)
```

2.10 Reviews with rating 1 or 2

Filter reviews first and join to the borrower and book.

```
Answer = π_{BORROWER.CardNo, BORROWER.Name,
           BOOK.Title, BOOK_REVIEWS.Rating} (
  BORROWER ⋈ σ_{Rating = 1 OR Rating = 2}(BOOK_REVIEWS) ⋈ BOOK
)
```

note: Projection removes duplicates under relational-algebra set semantics.

2.11 Staff at branches owning wishlisted books

Require a positive copy count and a matching wishlist book.

```
WishlistedBooks = π_{BookID}(NEW_BOOK_WISHLIST)
EligibleBranches = π_{BranchID} (
  σ_{NumCopies > 0}(BOOK_COPIES ⋈ WishlistedBooks)
)
BranchInfo = π_{BranchID, BranchName}(LIBRARY_BRANCH)
Answer = π_{LIBRARY_STAFF.EmployeeID, LIBRARY_STAFF.Name,
           BranchName} (
  LIBRARY_STAFF ⋈ EligibleBranches ⋈ BranchInfo
)
```

note: A BOOK_COPIES row with NumCopies = 0 does not establish ownership.

2.12 Borrowers who checked out every Charles Village book

Relational division checks that each borrower borrowed every available book at that branch.

```
CharlesVillageBooks = π_{BookID} (
  σ_{BranchName = 'Charles Village' AND NumCopies > 0} (
```

```
    BOOK_COPIES ⋈ LIBRARY_BRANCH
  )
)
BorrowedBooks = π_{CardNo, BookID}(BOOK_LOANS)
EligibleCardNumbers = BorrowedBooks ÷ CharlesVillageBooks
Answer = π_{CardNo, Name}(BORROWER ⋈ EligibleCardNumbers)
```

note: The loans need not originate at Charles Village; only the set of BookIDs available there matters. If Charles Village had no available books, the division would return every CardNo that appears in BOOK_LOANS.

2.13 Compare copies of two books at the same branch

Get each book's per-branch count, then join the two results on BranchID and compare.

```
DishwashingCopies = ρ_{DishwashingCopies} (
  π_{BranchID, NumCopies} (
    σ_{Title = 'Dishwashing for Dummies'}(BOOK ⋈ BOOK_COPIES)
  )
)
BraveNewWorldCopies = ρ_{BraveNewWorldCopies} (
  π_{BranchID, NumCopies} (
    σ_{Title = 'Brave New World'}(BOOK ⋈ BOOK_COPIES)
  )
)
EligibleBranches = ρ_{BranchID ← DishwashingCopies.BranchID} (
  π_{DishwashingCopies.BranchID} (
    σ_{DishwashingCopies.BranchID = BraveNewWorldCopies.BranchID
      AND DishwashingCopies.NumCopies > BraveNewWorldCopies.NumCopies} (
      DishwashingCopies × BraveNewWorldCopies
    )
  )
)
Answer = π_{BranchName, Address}(LIBRARY_BRANCH ⋈ EligibleBranches)
```

2.14 Books with multiple authors

Find two distinct author names associated with one BookID.

```
FirstAuthor = ρ_{FirstAuthor}(BOOK_AUTHORS)
SecondAuthor = ρ_{SecondAuthor}(BOOK_AUTHORS)
MultipleAuthorBooks = ρ_{BookID ← FirstAuthor.BookID} (
  π_{FirstAuthor.BookID} (
    σ_{FirstAuthor.BookID = SecondAuthor.BookID AND
      FirstAuthor.AuthorName ≠ SecondAuthor.AuthorName} (
      FirstAuthor × SecondAuthor
    )
  )
)
Answer = π_{Title, PublisherName}(BOOK ⋈ MultipleAuthorBooks)
```

2.15 Available at Charles Village, not Towson South

Availability means at least one copy, and the set difference is by BookID.

```
CharlesVillageBooks = π_{BookID} (
  σ_{BranchName = 'Charles Village' AND NumCopies > 0} (
    BOOK_COPIES ⋈ LIBRARY_BRANCH
  )
)
TowsonSouthBooks = π_{BookID} (
  σ_{BranchName = 'Towson South' AND NumCopies > 0} (
    BOOK_COPIES ⋈ LIBRARY_BRANCH
  )
)
EligibleBooks = CharlesVillageBooks − TowsonSouthBooks
Answer = π_{Title, PublisherName}(BOOK ⋈ EligibleBooks)
```

2.16 Borrowers without a loan from a branch in their city

Subtract borrowers who have any loan from a branch whose city equals their own.

```
AllBorrowers = π_{CardNo, Name, Address}(BORROWER)
LocalBorrowers = π_{BORROWER.CardNo, BORROWER.Name,
                    BORROWER.Address} (
  σ_{BORROWER.City = LIBRARY_BRANCH.City} (
    BORROWER ⋈_{BORROWER.CardNo = BOOK_LOANS.CardNo} BOOK_LOANS
    ⋈_{BOOK_LOANS.BranchID = LIBRARY_BRANCH.BranchID} LIBRARY_BRANCH
  )
)
Answer = AllBorrowers − LocalBorrowers
```

note: Borrowers with no loans anywhere also qualify.

2.17 Borrowed from every library branch

Divide the borrower–branch loan pairs by all library branches.

```
BorrowerBranches = π_{CardNo, BranchID}(BOOK_LOANS)
AllBranches = π_{BranchID}(LIBRARY_BRANCH)
EligibleCardNumbers = BorrowerBranches ÷ AllBranches
Answer = π_{Name, Address, CardNo}(BORROWER ⋈ EligibleCardNumbers)
```

note: Requires at least one loan at each branch, not one book borrowed at every branch.

Part B — Tuple Relational Calculus

The notation "∃ staff ∈ LIBRARY_STAFF" abbreviates "there exists a tuple staff belonging to LIBRARY_STAFF." Each t denotes an output tuple with the requested named fields.

2.2 Staff working in Kim Kardashian's city

Match a borrower city to the city of the branch employing each staff member.

```
{ t | ∃ staff ∈ LIBRARY_STAFF, ∃ branch ∈ LIBRARY_BRANCH,
      ∃ borrower ∈ BORROWER :
    borrower.Name = 'Kim Kardashian'
    AND borrower.City = branch.City
    AND staff.BranchID = branch.BranchID
    AND t.Name = staff.Name
    AND t.Position = staff.Position }
```

2.6 Staff at branches where Steve Bannon has a loan

Match Steve Bannon to a loan and match that loan's branch with each staff member's branch.

```
{ t | ∃ staff ∈ LIBRARY_STAFF, ∃ borrower ∈ BORROWER,
      ∃ loan ∈ BOOK_LOANS :
    borrower.Name = 'Steve Bannon'
    AND borrower.CardNo = loan.CardNo
    AND staff.BranchID = loan.BranchID
    AND t.Name = staff.Name
    AND t.Position = staff.Position
    AND t.Phone = staff.Phone }
```

2.7 Individual rooms at both branches on same date

The same borrower must have two reservations, at both named branches, for individual rooms on one date.

```
{ t | ∃ borrower ∈ BORROWER,
      ∃ firstReservation ∈ ROOM_RESERVATION,
      ∃ secondReservation ∈ ROOM_RESERVATION,
      ∃ firstRoom ∈ MEETING_ROOM, ∃ secondRoom ∈ MEETING_ROOM,
      ∃ firstBranch ∈ LIBRARY_BRANCH,
      ∃ secondBranch ∈ LIBRARY_BRANCH :
    borrower.CardNo = firstReservation.CardNo
    AND borrower.CardNo = secondReservation.CardNo
    AND firstReservation.ReservationDate = secondReservation.ReservationDate
    AND firstReservation.RoomID = firstRoom.RoomID
    AND secondReservation.RoomID = secondRoom.RoomID
    AND firstRoom.RoomType = 'Individual'
    AND secondRoom.RoomType = 'Individual'
    AND firstRoom.BranchID = firstBranch.BranchID
    AND secondRoom.BranchID = secondBranch.BranchID
    AND firstBranch.BranchName = 'Charles Village'
    AND secondBranch.BranchName = 'Towson South'
    AND t.Name = borrower.Name }
```

note:

- The two reservations do not need a separate inequality condition; the branch names ‘Charles Village’ and ‘Towson South’ differ, so they are necessarily different reservations.
- Both rooms must be of type ‘Individual’, at the two named branches, on the same ReservationDate.
- BORROWER has no Phone attribute, so only Name is returned.

2.8 Borrowers who reviewed multiple distinct books

Require two reviews from the same borrower involving different book IDs.

```
{ t | ∃ borrower ∈ BORROWER,
      ∃ firstReview ∈ BOOK_REVIEWS,
      ∃ secondReview ∈ BOOK_REVIEWS :
    borrower.CardNo = firstReview.CardNo
    AND borrower.CardNo = secondReview.CardNo
    AND firstReview.BookID ≠ secondReview.BookID
    AND t.Name = borrower.Name }
```