COMP-SCI 5570: Architecture of Database Management Systems
Case Study: Library Loan Record Tracking System

Final Project Report

Bindu Parvati Jonnala Gadda
16338568

Develop a library loan record tracking system to track loan records in a library. Library patrons may borrow books, magazines, movies, compact discs, and audio tapes. Each copy of a library item must be managed. For example, a library may have five copies of the book "The Grapes of Wrath". Your system must track borrowers, their addresses, and phone number. Each borrower is assigned a library card.

Each category of library item has a standard checkout period and number of renewals. For example, children's books may be checked out for a month, while adult books may be checked out for only two weeks. Ordinary books can be renewed once; books with a pending request may not be renewed. Your system must record the actual return date and any payments that was paid. Your system must maintain any records of damage and loss and associated payments. Your system must also enable searches, such as

• Total payments owed by a patron.
• payments revenue to the library for an interval of time.
• The copies of library items that are grossly overdue. \Grossly" is the number of days specified by the librarian.

Develop a logical data model based on the following requirements:

• Refinement of the conceptual model - including a refined Enhanced Entity-Relationship (EER) diagram.
• Derivation of relations from the refined conceptual model.
• Validation of logical model using normalization to BCNF.
• Validation of logical model against corresponding user transactions.
• Definition of integrity constraints including:
    1. Primary key constraints.
    2. Referential Integrity (foreign key) constraints.
    3. Entity integrity (NULL and default value) constraints.
    4. Alternate key constraints.
    5. General constraints if any.

Identification of Entities:

- LibraryItem
- Borrower
- ItemCategory
- ItemCopies
- FineDetails
- ItemOperations
- PaymentDetails

Attributes associated with the Entities:

- LibraryItem(ItemID, ItemName, ItemDetails)
- Borrower(BorrowerID, BorrowerName, PhoneNo, Address)
- ItemCategory(CategoryID, CategoryName, CheckoutPeriod, NoOfRenewals)
- ItemCopies(NoOfCopies)- This is a weak entity
- FineDetails(FineID, FineDate, FineAmount)
- PaymentDetails(BorrowerID, FineID, ItemID, PaymentDate, PaymentType)
- ItemOperations(ItemID, BorrowerID, CheckedOutOn, RenewedOn, PendingRequest)

Assumptions:
ItemID is unique for different books. Each item available in the library has it's own ID.
Each item should belong to only one type and one category.

Relationships among entities:

1. LibraryItem includes one ItemTypes and each ItemTypes can be included by one or more items.
2. Each LibraryItem may belong to atleast one category and each category can have one or more LibraryItems.
3. Each LibraryItem can have one or more ItemCopies, but each copy belongs to LibraryItem.
4. Fine may consist more than one LibraryItem, but each LibraryItem may be involved in zero or more Payments. Each item may have different expectedReturnDate, ReturnedDate, periodOfRenewal, isOverdue.

5. Each borrower may have one or more fines, but all the fines should belong to the same borrower.
6. Each ItemCopy may or may not have an ItemOperations, but all the ItemOperations should belong to an ItemCopy.
7. Each Borrower may or may not have PaymentDetails, but each payment detail should belong to atleast one borrower.

Participation and Cardinality Constraints:

LibraryItem includes one ItemTypes and each ItemTypes can be included by one or more items.

- Cardinality of LibraryItem to ItemType is one to many because there will be more than one ItemType. Participation is optional because all the ItemTypes need not belong to one LibraryItem.

Each LibraryItem may belong to atleast one category and each category can have one or more LibraryItems.

- Cardinality of LibraryItem to ItemCategory is one to many because there will be more than one ItemCategory. Participation is optional for LibraryCategory because it need not have LibraryItem. Participation is mandatory for LibraryItem because each item should belong to atleast one category.

Each LibraryItem consists one or more ItemCopies, but each copy belongs to atleast one LibraryItem

- Cardinality in this relation is one to many because each item can have more than one ItemCopies. Participation is mandatory because each item should have a copy and each ItemCopy should considered as LibraryItem.

Fine may consist more than one LibraryItem, but each LibraryItem may be involved in zero or more Payments. Each item may have different expectedReturnDate, ReturnedDate, periodOfRenewal, isOverdue.

- Cardinality is many to many Each fine may belong to one or more copies one copy of LibraryItem and each copy of LibraryItem

Each borrower may have one or more fines, but all the fines should belong to the same borrowerr.

- Cardinality in this relation is one to many because each borrower may have one or more fines to be paid but a fine is related to one borrower only. Participation is mandatory because each borrower should be fined for violation and each fine should belong to a borrower.

Each ItemCopy may or may not have an ItemOperations, but all the ItemOperations should belong to an ItemCopy.

- Cardinality of this relation is many to one because each itemcopy can have one or more operations. Participation is optional for every itemcopy in itemoperations and mandatory for item operations in itemcopy.

Each Borrower may or may not have PaymentDetails, but each payment detail should belong to atleast one borrower.

- Cardinality of this relation is one to many because, each borrower can have one or more payments, while each payment should belong to atleast one borrower. Participation of borrower is mandatory for payments while paymentdetails are optional for each borrower.

Determination of Primary Key and Candidate Key:

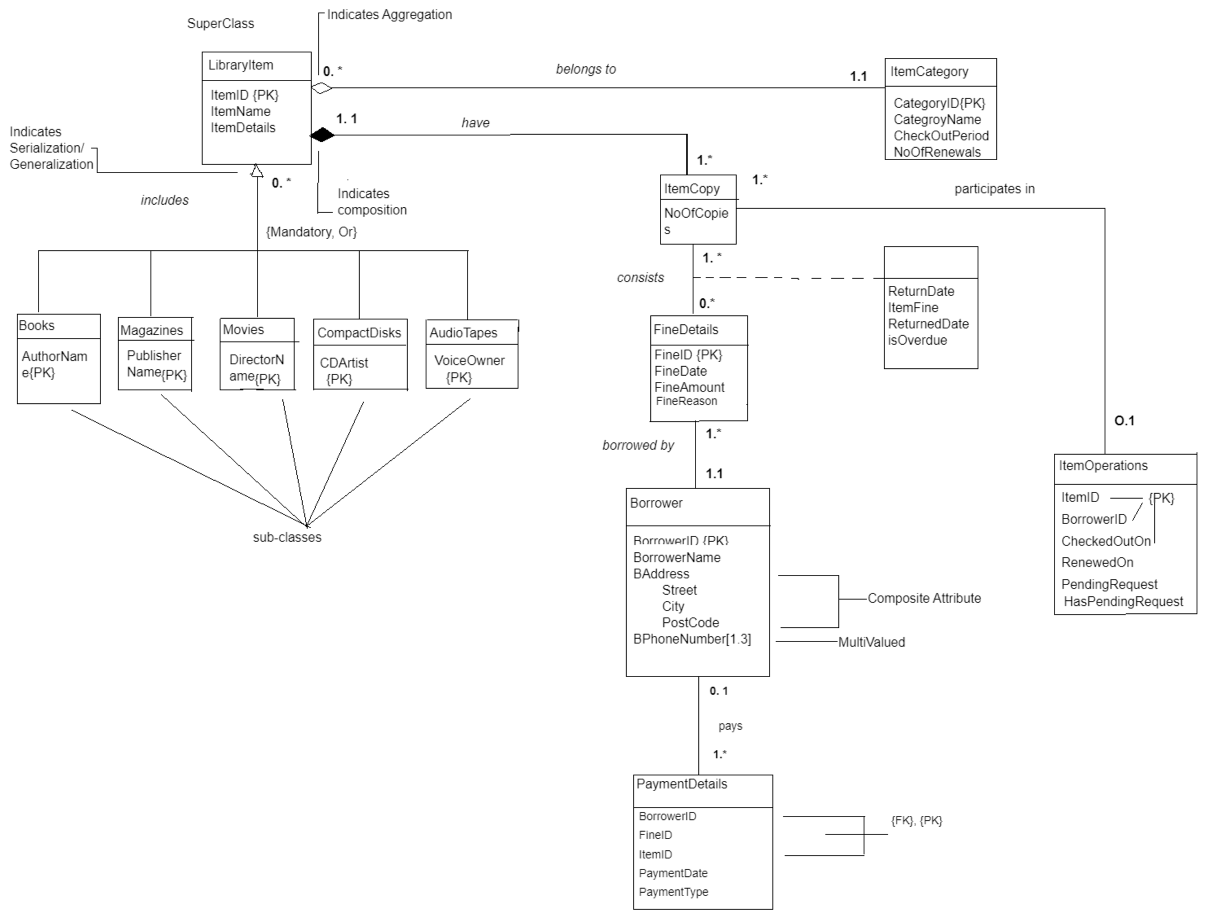
LibraryItem(<u>ItemID</u>, ItemName, ItemDetails)- ItemID is primary key
Each of the types are considered as subclass for the superclass LibraryItem
Books(<u>AuthorName)</u>
Magazines(<u>PublisherName)</u>
Movies(<u>DirectorName)</u>
CompactDisks(<u>CDArtist</u>)
AudioTapes(<u>Voice_Owner)</u>
Borrower(<u>BorrowerID</u>, BorrowerName, PhoneNo, Address)- BorrowerID is primary key
ItemCategory(<u>CategoryID</u>, CategoryName, CheckoutPeriod, NoOfRenewals)- CategoryID is primary key. Uniquely identifies the relation
ItemCopies(NoOfCopies) - Weak entity hence it doesn't have primary key
FineDetails(<u>FineID</u>, FineDate, FineAmount)- FineID is primary key
ItemOperations(<u>ItemID, BorrowerID</u>, CheckedOutOn, RenewedOn, PendingRequest)
PaymentDetails(BorrowerID, <u>FineID, ItemID</u>, PaymentDate, PaymentType)

Determination of specialization/generalization and categorization relationships:

Considering LibraryItem is a super class which has attributes called ItemID, ItemName, ItemDetails. There are 5 sub classes for the same which inherits all the properties of superclass LibraryItem. They are Books, Magazines, Movies, Compact discs, and Audio tapes. Respective attributes are AuthorName, PublisherName, DirectorName, CDArtist, VoiceOwner.
Constraints on specialization/generalization:
Mandatory participation is needed for every subclass. Disjoint is Or because one item can belong to one type only as per my assumption listed above.

Categorization:
Aggregation : LibraryItem is whole, ItemCategory, ItemCopy are parts.
ItemCopy has composition .Each ItemCopy belongs to exactly one LibraryItem entity

-> Enhanced Entity Relationship Diagram:



->Derivation of relations from the refined conceptual model:

We describe how relations are derived for the following structures that may occur in a conceptual data model:

(1) **strong entity types:**

- LibraryItem(ItemID, ItemName, ItemDetails), ItemID is **primary key**
- Borrower(BorrowerID, BorrowerName, BPhoneNumber, BStreet, BCity,BPostCode)- BorrowerID is **primary key**
- ItemCategory(CategoryID, CategoryName, CheckoutPeriod, NoOfRenewals)- CategoryID is **primary key**. Uniquely identifies the relation
- FineDetails(FineID, FineDate, FineAmount)- FineID is **primary key**

(2) **weak entity types**:

- ItemCopies(NoOfCopies) - Weak entity hence it doesn't have primary key

(3) **one-to-many (1:*) binary relationship types:**

- ItemCopies(ItemID, NoOfCopiesAvailable, CopiesBorrowed) - ItemID is foreign key references LibraryItem(ItemID)
- LibraryItem(ItemID, ItemName, ItemDetails, CategoryID), ItemID is primary key, CategoryID is foreign key references ItemCategory(CategoryID)
- FineDetails(FineID, FineDate, FineAmount, BorrowerID)- FineID is primary key, BorrowerID is foreign key references Borrower(BorrowerID)

(4) **superclass/subclass relationship types:**

- Books(AuthorName, ItemID) - AuthorName is primary key, ItemID is foreign key references LibraryItem(ItemID)
- Magazines(PublisherName, ItemID) - PublisherName is primary key, ItemID is foreign key references LibraryItem(ItemID)
- Movies(DirectorName, ItemID) - DirectorName is primary key, ItemID is foreign key references LibraryItem(ItemID)
- CompactDisks(CDArtist, ItemID) - CDArtist is primary key, ItemID is foreign key references LibraryItem(ItemID)
- AudioTapes(VoiceOwner, ItemID) - VoiceOwner is primary key, ItemID is foreign key references LibraryItem(ItemID)

(5) **many-to-many (*:*) binary relationship types:**

- Returns(ItemID, FineID, ReturnDate, ItemFine, ReturnedDate, isOverdue) - ItemID, FineID forms primary key, ItemID is foreign key references LibraryItem(ItemID), FineID is foreign key references FineDetails(FineID)

(6) **multi-valued attributes:**

- Telephone(BPhoneNumber, BorrowerID) - BPhoneNumber is the primarykey and BorrowerID is foreign key references Borrower(BorrowerID)

->Validation of logical model using normalization to BCNF.

LibraryItem(ItemID, ItemName, ItemDetails, CategoryID)

No non- candidatekey is determinant in this relation, hence it is in BCNF
o

- Borrower(BorrowerID, BorrowerName, BPhoneNumber, BStreet, BCity,BPostCode)

Here BStreet-> BCity, BCity->BPostCode which is a transitive dependency.
Assuming the database is being designed for a public library loan record tracking belongs to a particular city.

- Borrower(BorrowerID, BorrowerName, BPhoneNumber, BStreet)

Address(BStreet, BCity, BPostCode)- BStreet is the foreign key of Borrower(BStreet)
This is after 3NF, still the non-candidatekey BorrowerName, Bstreet exists, to remove this and convert the relation to BCNF

- Borrower(BorrowerID, BorrowerName)
- BorrowerDetails(BorrowerID, BPhoneNumber, BStreet)
- Address(BorrowerID, BCity, BPostCode)BStreet is the foreign key of BorrowerDetails(BStreet)

ItemCategory(CategoryID, CategoryName, CheckoutPeriod, NoOfRenewals)
No non- candidatekey is determinant in this relation, hence it is in BCNF

FineDetails(FineID, FineDate, FineAmount, BorrowerID)
No non- candidatekey is determinant in this relation, hence it is in BCNF

Returns(ItemID, FineID, ReturnDate, ItemFine, ReturnedDate, isOverdue)
Here ReturnDate->ReturnedDate, ReturnedDate->isOverdue which is a transitive dependency.
Assuming isOverdue is set to true only if there is difference between ReturnDate and ReturnedDate.
Returns(ItemID, FineID, ItemFine)
FineConfirmation(FineID, ReturnDate, ReturnedDate, isOverdue) FineID is the foreignkey of Returns(FineID)

Telephone(BPhoneNumber, BorrowerID)
No non- candidatekey is determinant in this relation, hence it is in BCNF

ItemCopies(ItemID, NoOfCopiesAvailable, CopiesBorrowed)
No non- candidatekey is determinant in this relation, hence it is in BCNF

-> Validation of logical model against corresponding user transactions:

Validating the given below user transactions,

• Total fine owed by a patron.
To find out the total fine owned by a patron/borrower performing the sum of FineAmount of FineDetails relation for a corresponding borrowerID.

• Fine revenue to the library for an interval of time.
To find the revenue of library over a period of time, FineDate attribute of FineDetails can be used and sum of all FineAmount in the period helps to find it out.

• The copies of library items that are grossly overdue. \Grossly" is the number of days specified by the librarian.
The items which has difference between ReturnDate and ReturnedDate where isOverdue is true can be fetched from the Returns relation.

-> Definition of integrity constraints including:

Each column should contain atomic values
All the attributes belonging to one column in a relation should have same domain

- Primary key constraints: All the primary keys of the relations mentioned above should have unique values and cannot have null values .

  For example, In the relation LibraryItem(<u>ItemID</u>, ItemName, ItemDetails, CategoryID) where ItemID is primary key, CategoryID is foreign key references ItemCategory(CategoryID).

  Here ItemID will have unique values and cannot have null values. Similarly all the primary keys in all the relations should follow the same. NOT NULL keyword will be used while writing queries.

- Referential Integrity (foreign key) constraints: If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be wholly null.

  All the relations having foreign key will implement this constraint.

  For example, In the relation LibraryItem(<u>ItemID</u>, ItemName, ItemDetails, CategoryID) where ItemID is primary key, CategoryID is foreign key references ItemCategory(CategoryID). Here the CategoryID is foreign key which will either have a candidate key value of some tuple in ItemCategory or it can be wholly null.

- Entity integrity (NULL and default value) constraints: This states that no attribute of a primary key can be null.

The relations designed will make sure that there will be no null value in primary key attributes.

For example, In the relation LibraryItem(ItemID, ItemName, ItemDetails, CategoryID) where ItemID is primary key, here ItemID cannot be null. Similarly all the primary keys in all the relations should follow the same. NOT NULL keyword will be used while writing queries.

- General constraints if any:

Each borrower can lend only 5 items at a time, to take 6th, one of the itemshould be returned.

Creating and inserting values into Tables:

- ItemCategory(CategoryID, CategoryName, CheckoutPeriod, NoOfRenewals)

CREATE TABLE ItemCategory(CategoryID VARCHAR(20) Primary Key NOT NULL, CategoryName VARCHAR(100), CheckoutPeriod NUMERIC(5) NOT NULL, NoOfRenewals NUMERIC(5))

INSERT INTO ItemCategory(CategoryID, CategoryName, CheckOutPeriod, NoOfRenewals)
VALUES('K1', 'Kids', 30, 2),
('T1', 'Teen', 25, 2),
('Y1', 'Young Adult', 20,1),
('A1', 'Adult', 15, 1),
('S1', 'Senior Citizens', 30,1);

- LibraryItem(ItemID, ItemName, ItemDetails, CategoryID), ItemID is primary key, CategoryID is foreign key references ItemCategory(CategoryID)

CREATE TABLE LibraryItem(ItemID VARCHAR(20) Primary Key NOT NULL, ItemName VARCHAR(100) NOT NULL, ItemDetails VARCHAR(100) NOT NULL, CategoryID VARCHAR(20) REFERENCES ItemCategory(CategoryID));

ALTER TABLE LibraryItem ADD CONSTRAINT no_action FOREIGN KEY(CategoryID) REFERENCES ItemCategory(CategoryID);

INSERT INTO libraryitem(ItemID, ItemName, ItemDetails, CategoryID)

```
VALUES(' A211', 'Tom and Jerry', 'A comic since 1980','K1' ),
('A26', 'Chota Bheem' , 'story of bravery', 'K1' ),
('A9843', 'Dragon Tales', 'Friendships lasts long', 'K1' ),
('B54151', 'Spider man', 'A brave man', 'T1'),
('B5612', 'Super Man', 'A fiction character', 'T1'),
('B1581', 'Journey to moon', 'A story', 'T1'),
('C5251', 'Harry Potter', 'A novel', 'Y1' ),
('C5452', 'Courage to soar', 'A novel', 'Y1' ),
('C9853', 'A girl from Africa', 'A novel', 'Y1' ),
('C1562', '', 'A novel', 'Y1' ),
('C1625', '', 'A novel', 'Y1' ),
('D9584', 'Money Heist', 'An adult novel', 'A1' ),
('D9524', 'Stranger things', 'An adult novel', 'A1' ),
('D1924', 'Dexter', 'An adult novel', 'A1' ),
('D8564', 'You are my sunshine after all the storms', 'Romantic comedy novel ',
'A1' ),
('G98525', 'Capitalism', 'Political drama', 'S1' ),
('G6575', 'Recession', 'Political drama', 'S1' ),
('G972', 'Frankie and Gracy', 'Two best friends', 'S1' );


UPDATE libraryitem SET itemname = 'Alladdin and his magic lamp' where
itemid ='C1562';
UPDATE libraryitem SET itemname = 'Tenali Ramakrishna' where itemid
='C1625';

INSERT INTO libraryitem(ItemID, ItemName, ItemDetails, CategoryID)
VALUES(' M9547', 'Times of India', 'Popular magazine of India','A1' ),
(' M2581', 'Panchathantra', 'Kids Weekly','K1' ),
(' M7516', 'Love what you do stories ', 'Young adults Weekly Magazine','Y1' ),
(' M3548', ' Political Drama ', ' Seniors Weekly Magazine','S1' ),
(' M7548', ' Young Rider ', ' Teens Weekly Magazine','T1' ),
(' M09547', 'A Train to Busan', 'Popular rated movie from Japan','A1' ),
(' M02581', 'Ramayana', 'Indian historical stories','K1' ),
(' M07516', ' Bahubali', 'A movie about ethics','Y1' ),
(' M03548', ' NTR Kathanayakudu', ' Historical movie','S1' ),
(' M07548', 'Harry potter ', ' A fictional drama','T1' ),
(' CD9547', 'Ramayana', 'CD for adults','A1' ),
(' CD2581', 'Panchathantra', 'CD for kids','K1' ),
(' CD7516', 'Mahabharatha ', ' CD for Young adults ','Y1' ),
(' CD3548', ' Spirituality', ' CD For Seniors ','S1' ),
(' CD7548', ' Bhagavadgeetha ', ' Teens audio tapes','T1' ),
('AT9547', 'Ramayana', 'adult audio tapes','A1' ),
(' AT2581', 'Panchathantra', 'kids audio tapes','K1' ),
(' AT7516', 'Mahabharatha ', ' Young adults audio tapes','Y1' ),
(' AT3548', ' Spirituality', ' Seniors audio tapes ','S1' ),
(' AT7548', ' Bhagavadgeetha ', ' Teens audio tapes','T1' ),
```

- ItemCopies(ItemID, NoOfCopiesAvailable, CopiesBorrowed)

  ```
  CREATE TABLE ItemCopies(ItemID VARCHAR(20) REFERENCES
  LibraryItem(ItemID), NoOfCopiesAvailable NUMERIC(10), CopiesBorrowed
  NUMERIC(10));
  ```

  ```
  INSERT INTO ItemCopies(ItemID,NoOfCopiesAvailable, CopiesBorrowed)
  VALUES(' A26',10, 7),
  ('B54151',5, 9),
  ('G6575',7, 6),
  ('D1924', 6, 3);
  ```

- Books(AuthorName, ItemID) - AuthorName is primary key, ItemID is foreign key references LibraryItem(ItemID)

  ```
  CREATE TABLE Books(AuthorName VARCHAR(100) Primary Key NOT NULL,
  ItemID VARCHAR(20), CONSTRAINT FK_Books FOREIGN KEY (ItemID)
  REFERENCES LibraryItem(ItemID) ON DELETE CASCADE);
  ```

  ```
  INSERT INTO Books( ItemId, AuthorName)
  Values(' A211', ' J K Rowling'),
  ('A26', ' Ram'),
  ('A9843', ' Prasad'),
  ('B54151',  ' Santhi'),
  ('B5612', ' Vani'),
  ('B1581', ' Yasaswini'),
  ('C5251', ' Sri'),
  ('C5452', ' Lekha'),
  ('C9853', ' Sam'),
  ('D9584',  ' Sandeep'),
  ('D9524', ' Chaitanya'),
  ('D1924',  ' Raji'),
  ```

('D8564', ' Hema'),
('G98525', ' Lekhana'),
('G6575', ' Ganesh'),
('G972', 'Lasya'),
('C1562', ' Chetan'),
('C1625', ' Shreya');

- Magazines(<u>PublisherName,</u> ItemID) - PublisherName is primary key, ItemID is foreign key references LibraryItem(ItemID)

  CREATE TABLE Magazines(PublisherName VARCHAR(100) Primary Key NOT NULL, ItemID VARCHAR(20), CONSTRAINT FK_Books FOREIGN KEY (ItemID) REFERENCES LibraryItem(ItemID) ON DELETE CASCADE);
  ALTER TABLE Magazines RENAME CONSTRAINTS FK_Books TO FK_Magazines

  INSERT INTO Magazines(PublisherName, ItemID)
  VALUES('Times of India', ' M9547'),
  ('Ramoji Films',' M2581'),
  ('Guide to life ', ' M7516'),
  (' Andhra Jyothi',' M3548'),
  (' Disney', ' M7548');

- Movies(<u>DirectorName,</u> ItemID) - DirectorName is primary key, ItemID is foreign key references LibraryItem(ItemID)

  CREATE TABLE Movies(DirectorName VARCHAR(100) Primary Key NOT NULL, ItemID VARCHAR(20), CONSTRAINT FK_Movies FOREIGN KEY (ItemID) REFERENCES LibraryItem(ItemID) ON DELETE CASCADE);

  INSERT INTO Movies(DirectorName, ItemID)
  VALUES('Rajamouli', ' Mo9547'),
  ('Vinayak',' Mo2581'),
  ('Puri Jagannath ', ' Mo7516'),
  (' Sujeeth',' Mo3548'),
  ('Trivikram', ' Mo7548');

- CompactDisks(CDArtist, ItemID) - CDArtist is primary key, ItemID is foreign key references LibraryItem(ItemID)

  CREATE TABLE CompactDisks(CDArtist VARCHAR(100) Primary Key NOT NULL, ItemID VARCHAR(20), CONSTRAINT FK_CDs FOREIGN KEY (ItemID) REFERENCES LibraryItem(ItemID) ON DELETE CASCADE);

  INSERT INTO CompactDisks(CDArtist, ItemID)
  VALUES ('Janaki', ' CD7516',),
  ('SP Balu', ' CD2581',),
  (' Chithra',' CD3548'),
  ('Shreya Ghoshal',' CD7548');

- AudioTapes(VoiceOwner, ItemID) - VoiceOwner is primary key, ItemID is foreign key references LibraryItem(ItemID)

  CREATE TABLE AudioTapes(VoiceOwner VARCHAR(100) Primary Key NOT NULL, ItemID VARCHAR(20), CONSTRAINT FK_ATs FOREIGN KEY (ItemID) REFERENCES LibraryItem(ItemID) ON DELETE CASCADE);

  INSERT INTO AudioTapes(ItemID, VoiceOwner)
  Values(' AT7548', ' HemaChandra'),
  ('AT9547', 'Bhargavi'),
  (' AT2581', 'Madhuri'),
  (' AT7516', 'Revanth'),
  (' AT3548', ' Arjith'),
  (' AT7548', ' Mohana');

- Borrower(BorrowerID, BorrowerName)

  CREATE TABLE Borrower(BorrowerID serial Primary Key NOT NULL, BorrowerName VARCHAR (100) NOT NULL);

  INSERT INTO Borrower(BorrowerID, BorrowerName)
  VALUES(1, 'Jack'),
  (2, 'Jane'),
  (34, 'Rose'),
  (21, 'Sam'),
  (12, 'Nithin');

- Telephone(BPhoneNumber, BorrowerID) - BPhoneNumber is the primarykey and BorrowerID is foreign key references Borrower(BorrowerID)

```
CREATE TABLE Telephone(BPhoneNumber NUMERIC(11) NOT NULL
UNIQUE Primary Key, BorrowerID serial, CONSTRAINT FK_TELEPHONE
FOREIGN KEY(BorrowerID) REFERENCES Borrower(BorrowerID) ON
UPDATE CASCADE);

INSERT INTO Telephone(BPhoneNumber, BorrowerID)
VALUES(5086540985, 1),
(1239873514, 2),
(9854672564,34),
(6135869875, 21),
(9136587456, 12);
```

- BorrowerDetails(BorrowerID, BPhoneNumber, BStreet)

```
CREATE TABLE BorrowerDetails( BorrowerID serial, CONSTRAINT
FK_BorrowerDetails FOREIGN KEY (BorrowerID) REFERENCES
Borrower(BorrowerID), BPhoneNumber serial, CONSTRAINT FK_BD
FOREIGN KEY(BPhoneNumber) REFERENCES Telephone(BPhoneNumber) ,
BStreet VARCHAR(50) NOT NULL);

ALTER TABLE BorrowerDetails
ALTER COLUMN bphonenumber TYPE NUMERIC(11);


INSERT INTO BorrowerDetails(BorrowerID, BPhoneNumber, BStreet)
VALUES(1, 5086540985, '365 W Plain Street'),
(2, 1239873514,  '105 E Armour Boulevard'),
(34, 9854672564,  '23 W Charlotte Street'),
(21, 6135869875,  '62 S kenneth Street'),
(12, 9136587456,  '12 N Pennsylvania Ave');
```

- Address(BorrowerID, BCity, BPostCode)BStreet is the foreign key of BorrowerDetails(BStreet)

```
CREATE TABLE Address(BorrowerID serial, CONSTRAINT
FK_BorrowerDetails FOREIGN KEY (BorrowerID) REFERENCES
Borrower(BorrowerID), BCity VARCHAR(50), BPostCode NUMERIC(6));
```

INSERT INTO Address(BorrowerID, BCity, BPostCode)
VALUES(1,' New York', 657419),
(2, 'Boston', 359515),
(34,' California', 388752),
(21, 'Kansas', 987544),
(12, 'San Antonio', 785461);

- FineDetails(<u>FineID</u>, FineDate, FineAmount, BorrowerID)- FineID is primary key, BorrowerID is foreign key references Borrower(BorrowerID)

  CREATE TABLE FineDetails(FineID serial NOT NULL Primary Key, FineDate DATE, FineAmount NUMERIC(50), BorrowerID serial, CONSTRAINT FK_BorrowerDetails FOREIGN KEY (BorrowerID) REFERENCES Borrower(BorrowerID));

  Considering fineAmount is same irrespective of number of days overdue.

  INSERT INTO FineDetails(FineID, FineDate, FineAmount, BorrowerID)
  VALUES(36587,'2022-08-15', 20, 1),
  (3265 ,'2018-08-30', 20, 1),
  (1254,'2022-10-10',20, 2),
  (8465,'2019-11-07',20, 21),
  (9842,'2022-07-09',20, 2),
  (5975,'2021-10-02',20, 34),
  (1684,'2020-04-14',200, 34),
  (1674,'2021-10-08',20, 12),
  (1984,'2022-05-15',20, 12),
  (3975, '2022-12-10', 20, 34);

  ALTER TABLE FineDetails ADD FineReason VARCHAR(100);

  INSERT INTO FineDetails(FineID, FineDate, FineAmount, BorrowerID)
  VALUES(6874,'2022-08-10', 20, 2);

- Returns(ItemID, FineID, ItemFine)

  CREATE TABLE Returns(ItemID VARCHAR(20), CONSTRAINT FK_Returns FOREIGN KEY (ItemID) REFERENCES LibraryItem(ItemID) , FineID serial REFERENCES FineDetails(FineID), ItemFine NUMERIC(15));

```
INSERT INTO Returns(ItemId, FineID, ItemFine)
VALUES('B54151',1254,20),
('B5612', 1674, 20),
('C1562', 1684,20),
('A26', 1984,10),
('C5251', 3265,30),
('C5452', 5975,30),
('C9853', 8465,30),
('D8564',9842,30),
('G972', 36587,15),
('B54151', 3975, 20);

INSERT INTO Returns(ItemId, FineID, ItemFine)
VALUES('D1924',6874,20);
```

- FineConfirmation(FineID, ReturnDate, ReturnedDate, isOverdue) FineID is the foreignkey of Returns(FineID)

```
CREATE TABLE FineConfirmation(
        FineID serial REFERENCES FineDetails(FineID),
        ReturnDate DATE NOT NULL, ReturnedDate DATE,
        isOverDue BOOLEAN GENERATED ALWAYS AS
(ReturnedDate>ReturnDate or ReturnedDate is NULL) STORED);


INSERT INTO FineConfirmation(FineID,  ReturnDate, ReturnedDate)
VALUES(1254,'2022-08-10', '2022-08-15'),
(3975, '2022-12-10', NULL);
```

- PaymentDetails(BorrowerID, FineID, ItemID, PaymentDate, PaymentType)

```
CREATE TABLE PaymentDetails(BorrowerID serial REFERENCES
Borrower(BorrowerID), FineID serial REFERENCES FineDetails(FineID),
ItemID VARCHAR(20), CONSTRAINT FK_PaymentDetails FOREIGN KEY
(ItemID) REFERENCES LibraryItem(ItemID), PaymentDate DATE,
PaymentType VARCHAR(20));

INSERT INTO PaymentDetails(BorrowerID, FineID, ItemID, PaymentDate,
PaymentType)
VALUES(2, 1254, 'B54151', '2022-10-20', 'Cash'),
(12, 1674,'B5612','2021-10-18', 'Credit card'),
(34,1684,'C1562', '2020-05-14', 'money order'),
(12, 1984, 'A26','2022-05-25', 'debit card'),
(1,3265,'C5251','2018-09-05', 'Paypal'),
(2,6874, 'D1924', '2022-08-20', 'cheque),'
(1, 36587, 'G972', '2022-08-18', 'Cash'),
(34,5975,'C5452', '2020-05-14', 'cash'),

INSERT INTO PaymentDetails(BorrowerID, FineID, ItemID, PaymentDate,
PaymentType)
VALUES(2,6874,'D1924', '2022-08-20', 'Cheque');
```

- ItemOperations(ItemID, BorrowerID, CheckedOutOn, RenewedOn, PendingRequest)

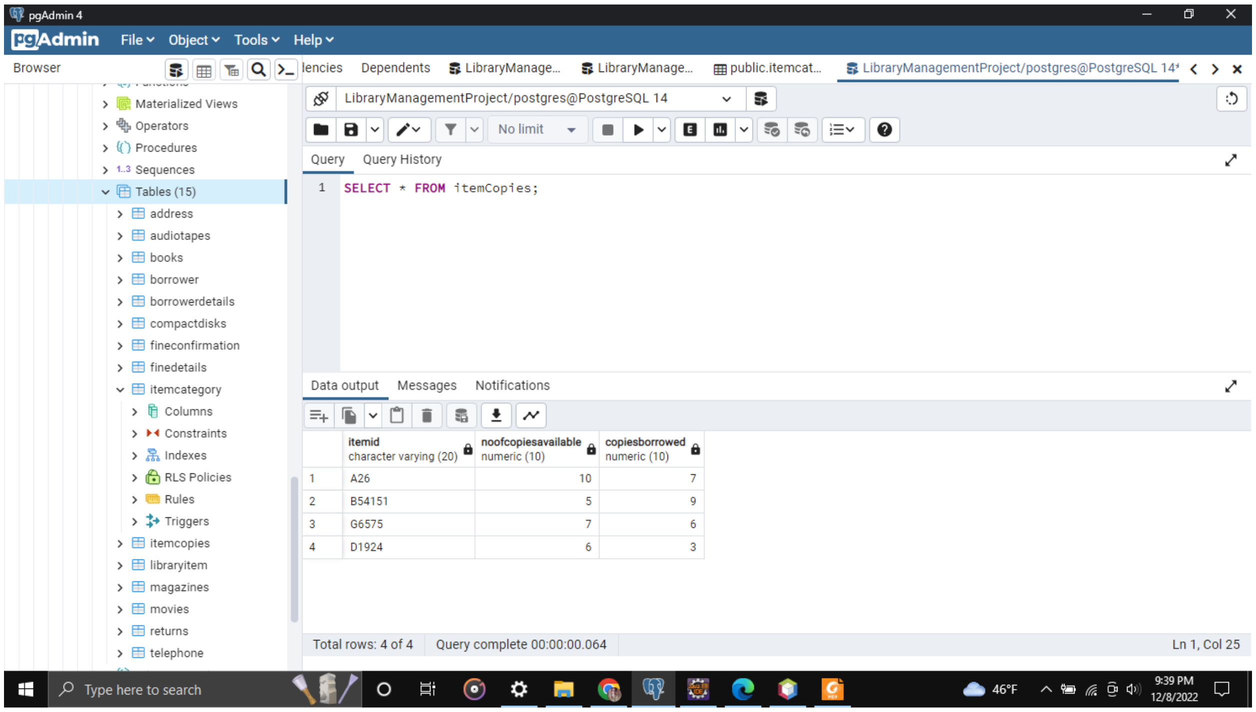
```
CREATE Table ItemOperations(ItemID VARCHAR(20),
CONSTRAINT FK_PaymentDetails FOREIGN KEY (ItemID) REFERENCES
LibraryItem(ItemID),
BorrowerID serial REFERENCES Borrower(BorrowerID),
CheckedOutOn DATE, RenewedOn DATE ,
PendingRequest serial,
canBeRenewed BOOLEAN GENERATED ALWAYS AS (PendingRequest Is
NULL) STORED );


INSERT INTO ItemOperations(ItemID, BorrowerID, CheckedOutOn,
RenewedOn, PendingRequest)
VALUES ('B54151', 2, '2022-07-16', ' ', ' '),
('A26', 2, '2022-09-10', NULL, ' 2022-09-14'),
('C5251', 34, '2019-09-10', '2019-09-30 ', NULL),
('C5452', 34, '2020-03-10', '2020-03-25 ', NULL),
('C1625', 34, '2020-04-10', '2020-04-25 ', NULL),
('C9853', 34, '2020-08-10', '2020-08-25 ', NULL),
('C1562', 34, '2021-011-10', '2021-011-25 ', NULL),
('D1924', 12, '2020-09-10', '2020-09-30 ', NULL),
('A26', 12, '2022-11-30', NULL, ' 2022-12-09'),
('G6575', 21, '2022-12-01', NULL, ' 2022-12-08'),
('D8564', 12, '2022-11-25', NULL, ' 2022-12-07');
```

1. List the number of copies of a particular library item.

The number of copies available for a particular item with ItemID A26 was listed using the query

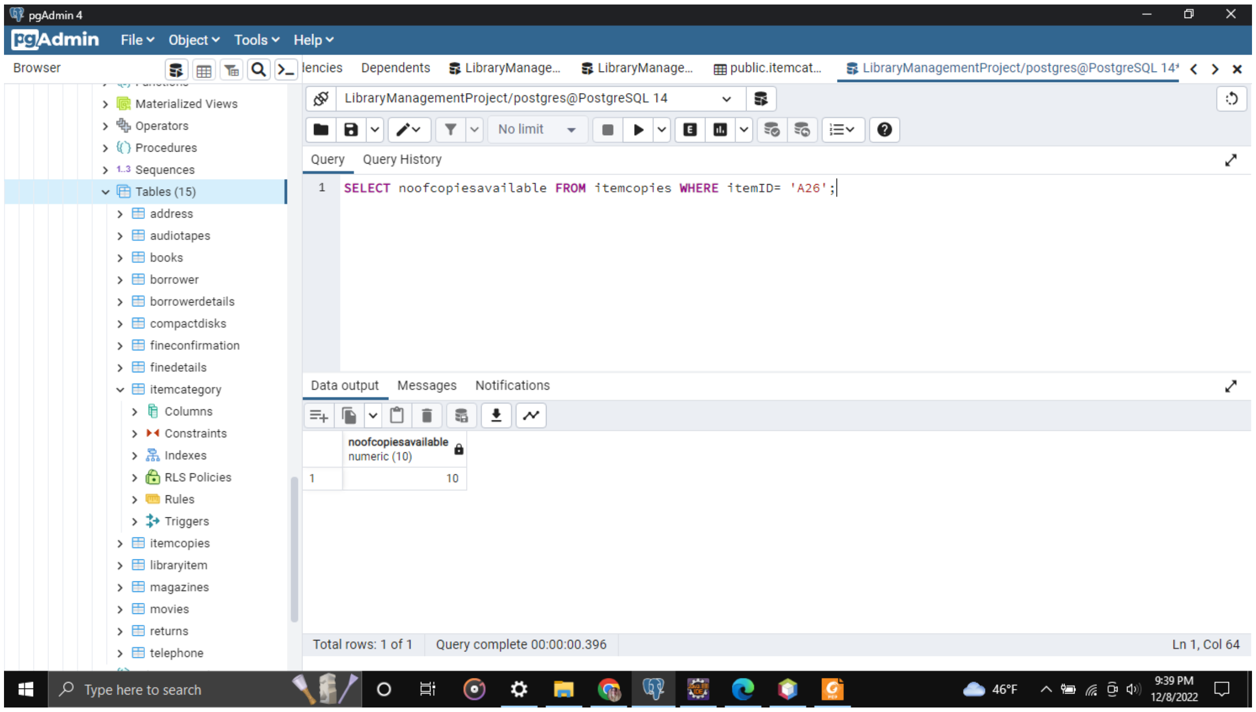
SELECT noofcopiesavaiable FROM itemcopies WHERE itemID= 'A26';



```
SELECT * FROM itemCopies;
```

	itemid character varying (20)	noofcopiesavailable numeric (10)	copiesborrowed numeric (10)
1	A26	10	7
2	B54151	5	9
3	G6575	7	6
4	D1924	6	3

Total rows: 4 of 4 Query complete 00:00:00.064 Ln 1, Col 25



```
SELECT noofcopiesavailable FROM itemcopies WHERE itemID= 'A26';
```

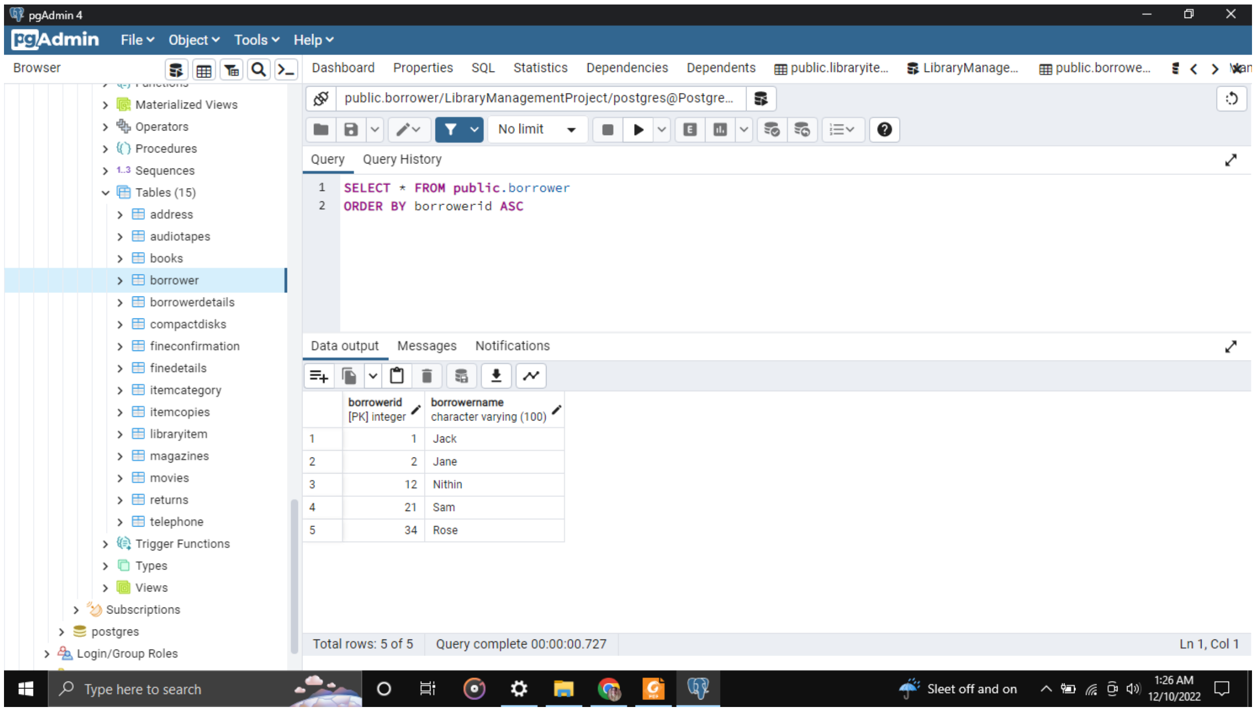
	noofcopiesavailable numeric (10)
1	10

Total rows: 1 of 1 Query complete 00:00:00.396 Ln 1, Col 64

2. List the details of the patrons who have at least an overdue library item today.

The details of patrons who have atleast an overdue library item today was listed by using the below query.

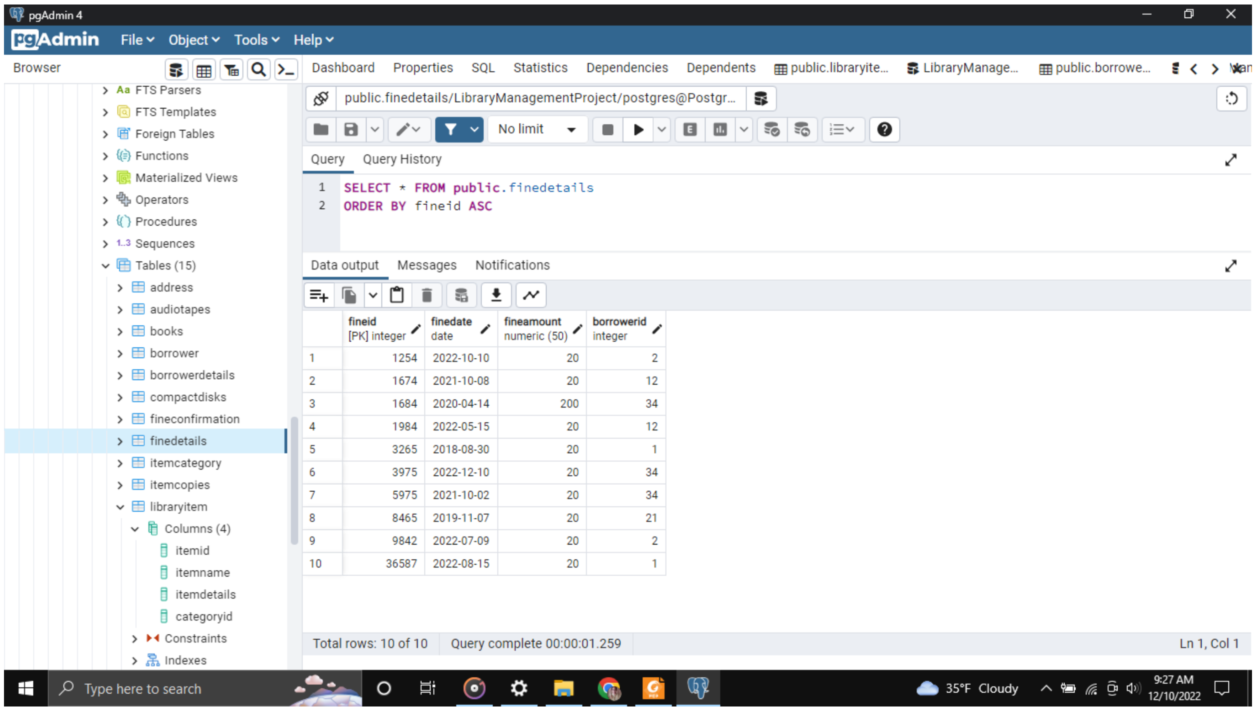
SELECT borrower.borrowerid, borrower.borrowername
FROM finedetails NATURAL JOIN returns
INNER JOIN borrower ON borrower.borrowerid = finedetails.borrowerid
INNER join fineconfirmation ON fineconfirmation.fineid = finedetails.fineid
WHERE fineconfirmation.returneddate is null ;



```
SELECT * FROM public.borrower
ORDER BY borrowerid ASC
```

	borrowerid [PK] integer	borrowername character varying (100)
1	1	Jack
2	2	Jane
3	12	Nithin
4	21	Sam
5	34	Rose

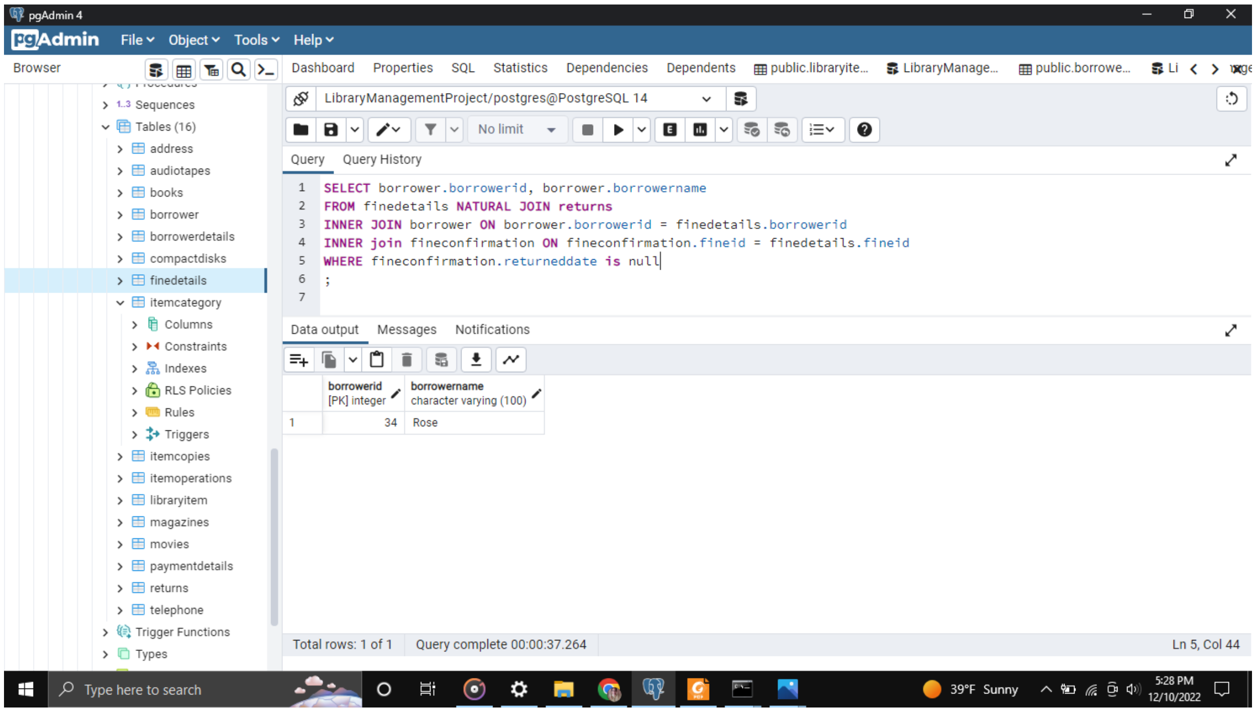
Total rows: 5 of 5   Query complete 00:00:00.727   Ln 1, Col 1



```
SELECT * FROM public.finedetails
ORDER BY fineid ASC
```

	fineid [PK] integer	finedate date	fineamount numeric (50)	borrowerid integer
1	1254	2022-10-10	20	2
2	1674	2021-10-08	20	12
3	1684	2020-04-14	200	34
4	1984	2022-05-15	20	12
5	3265	2018-08-30	20	1
6	3975	2022-12-10	20	34
7	5975	2021-10-02	20	34
8	8465	2019-11-07	20	21
9	9842	2022-07-09	20	2
10	36587	2022-08-15	20	1

Total rows: 10 of 10   Query complete 00:00:01.259   Ln 1, Col 1



```
SELECT borrower.borrowerid, borrower.borrowername
FROM finedetails NATURAL JOIN returns
INNER JOIN borrower ON borrower.borrowerid = finedetails.borrowerid
INNER join fineconfirmation ON fineconfirmation.fineid = finedetails.fineid
WHERE fineconfirmation.returneddate is null
;
```

	borrowerid [PK] integer	borrowername character varying (100)
1	34	Rose

Total rows: 1 of 1   Query complete 00:00:37.264   Ln 5, Col 44

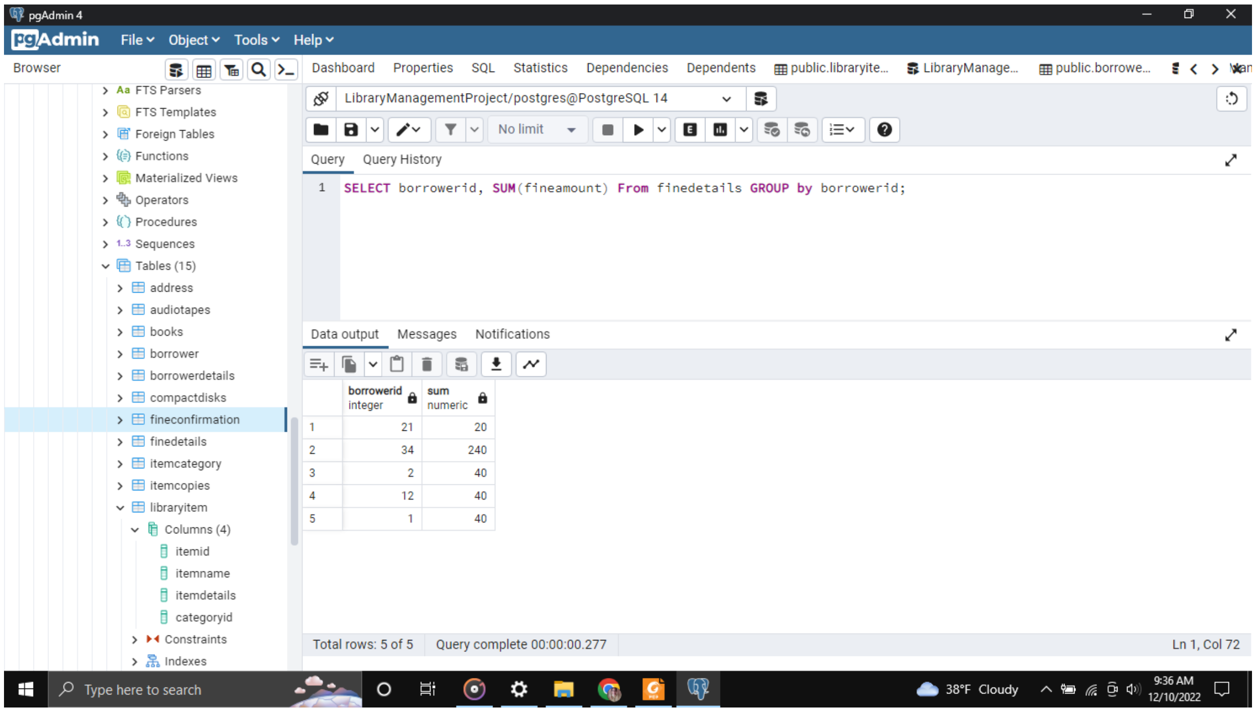
3. Identify the total fine owed by a patron (by his/her library card number) currently in the system.

BorrowerId is the library card number of a patron.

SELECT SUM(fineamount) FROM finedetails where borrowerId= 34;

SELECT sum(finedetails.fineamount) FROM finedetails left join paymentdetails on paymentdetails.fineid = finedetails.fineid where finedetails.borrowerId= 34 and paymentdetails.fineid is null;

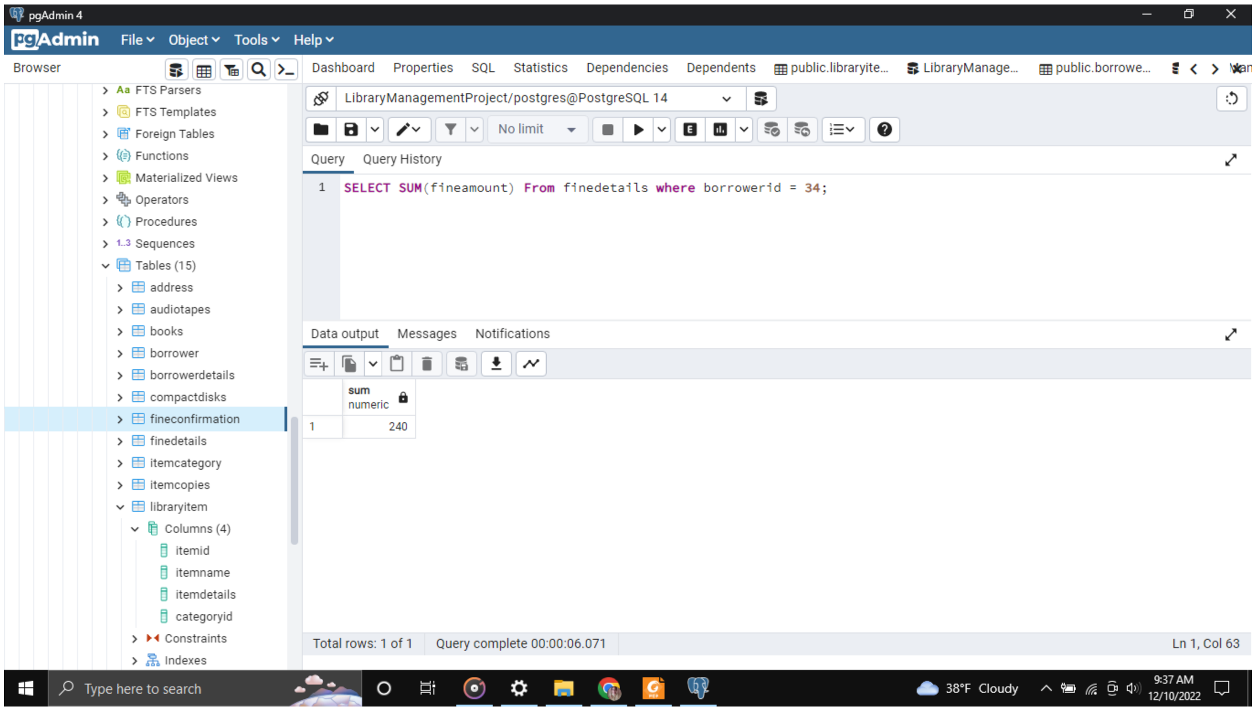
Total fine owned by a borrower with borrowerId = 34 is 240 out of which he made payments for 200. So the result of the operation to find the current due owed by borrower should be 40.



```
SELECT borrowerid, SUM(fineamount) From finedetails GROUP by borrowerid;
```

	borrowerid integer	sum numeric
1	21	20
2	34	240
3	2	40
4	12	40
5	1	40

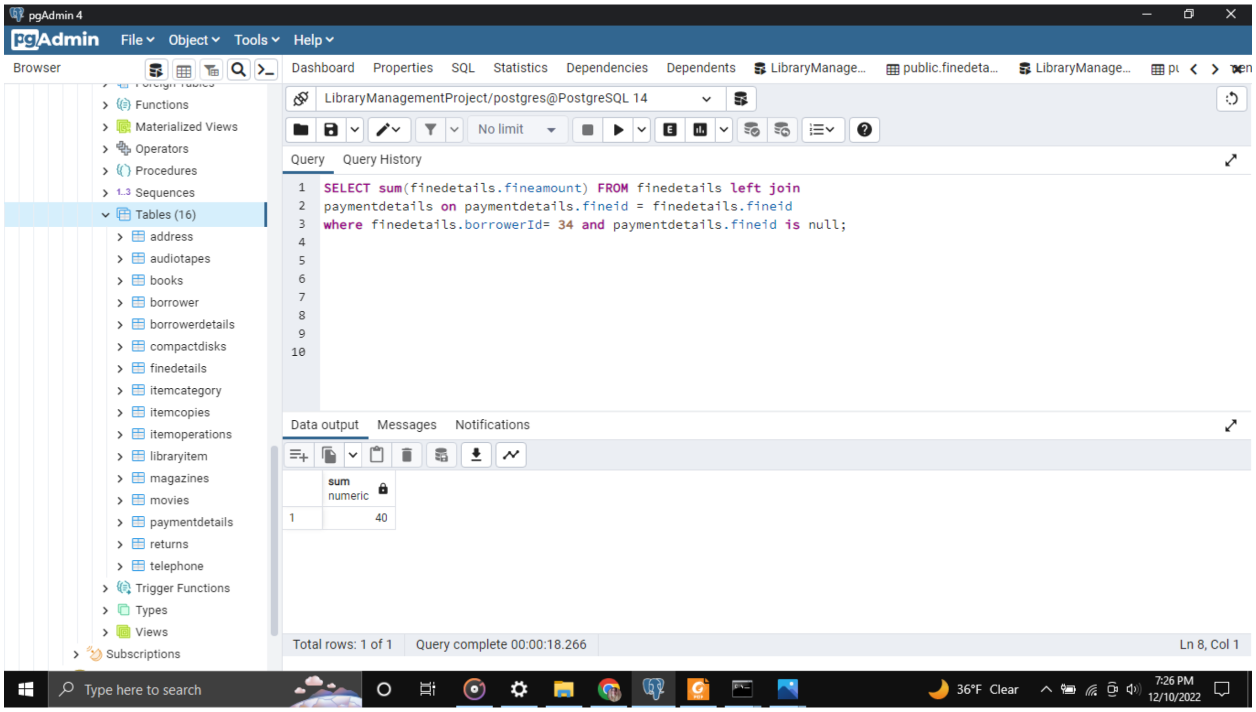
Total rows: 5 of 5   Query complete 00:00:00.277



```
SELECT SUM(fineamount) From finedetails where borrowerid = 34;
```

```
sum
numeric
1    240
```

Total rows: 1 of 1   Query complete 00:00:06.071   Ln 1, Col 63



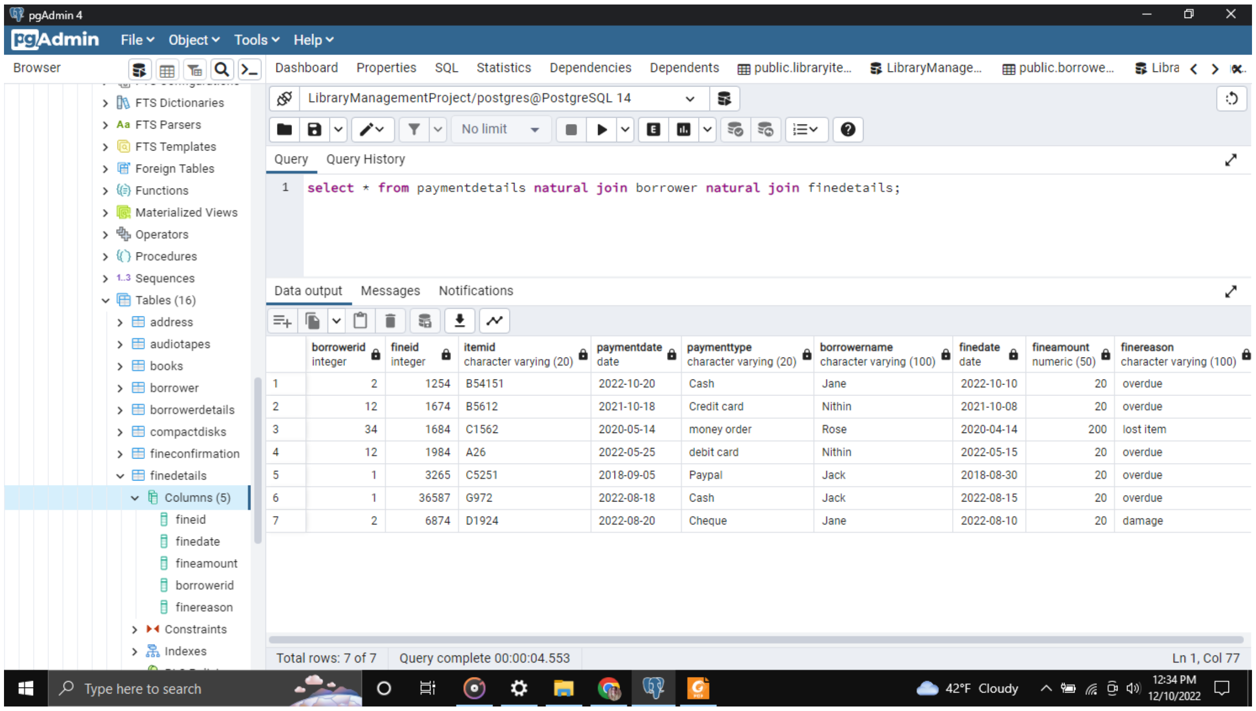
```
SELECT sum(finedetails.fineamount) FROM finedetails left join
paymentdetails on paymentdetails.fineid = finedetails.fineid
where finedetails.borrowerId= 34 and paymentdetails.fineid is null;
```

```
sum
numeric
1    40
```

Total rows: 1 of 1   Query complete 00:00:18.266   Ln 8, Col 1

4. List the details (e.g., damage, loss, amount, etc.) of the payment made by a patron.

*select * from paymentdetails natural join borrower natural join finedetails;*



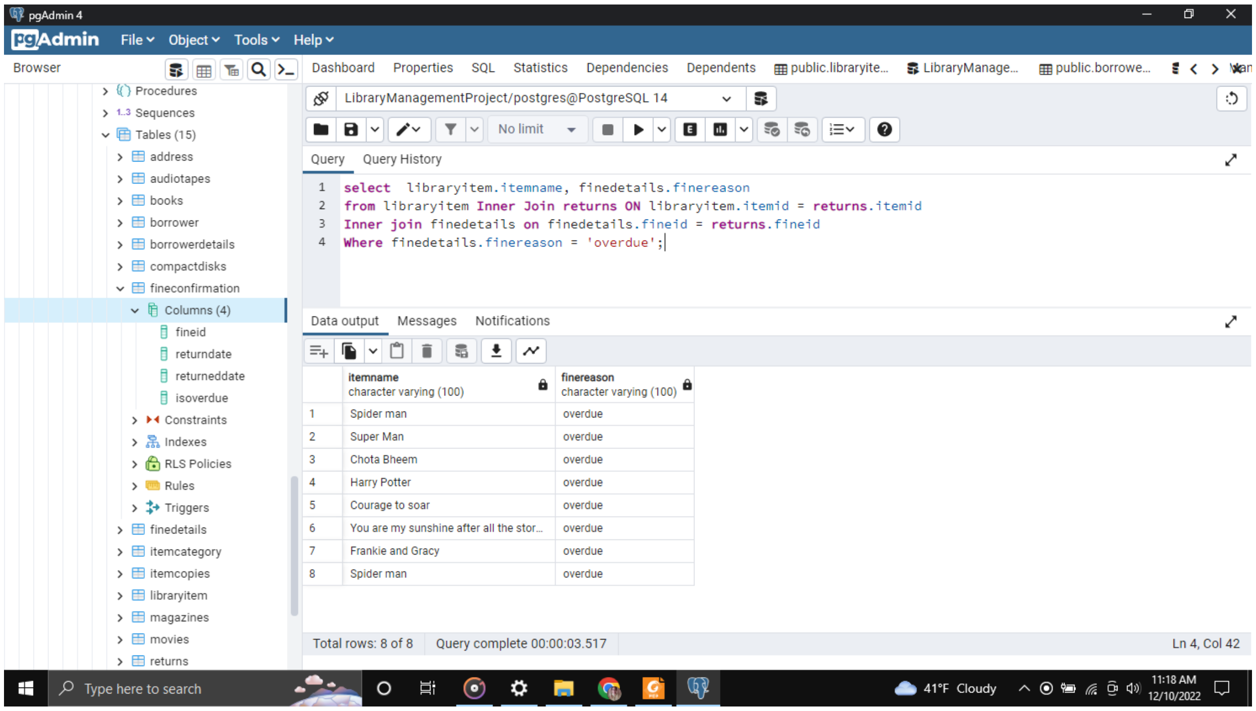
```
select * from paymentdetails natural join borrower natural join finedetails;
```

	borrowerid integer	fineid integer	itemid character varying (20)	paymentdate date	paymenttype character varying (20)	borrowername character varying (100)	finedate date	fineamount numeric (50)	finereason character varying (100)
1	2	1254	B54151	2022-10-20	Cash	Jane	2022-10-10	20	overdue
2	12	1674	B5612	2021-10-18	Credit card	Nithin	2021-10-08	20	overdue
3	34	1684	C1562	2020-05-14	money order	Rose	2020-04-14	200	lost item
4	12	1984	A26	2022-05-25	debit card	Nithin	2022-05-15	20	overdue
5	1	3265	C5251	2018-09-05	Paypal	Jack	2018-08-30	20	overdue
6	1	36587	G972	2022-08-18	Cash	Jack	2022-08-15	20	overdue
7	2	6874	D1924	2022-08-20	Cheque	Jane	2022-08-10	20	damage

Total rows: 7 of 7 Query complete 00:00:04.553 Ln 1, Col 77

5. List the copies of library items that are grossly overdue.

select libraryitem.itemname, finedetails.finereason
from libraryitem Inner Join returns ON libraryitem.itemid = returns.itemid
Inner join finedetails on finedetails.fineid = returns.fineid
Where finedetails.finereason = 'overdue';



```sql
select  libraryitem.itemname, finedetails.finereason
from libraryitem Inner Join returns ON libraryitem.itemid = returns.itemid
Inner join finedetails on finedetails.fineid = returns.fineid
Where finedetails.finereason = 'overdue';
```

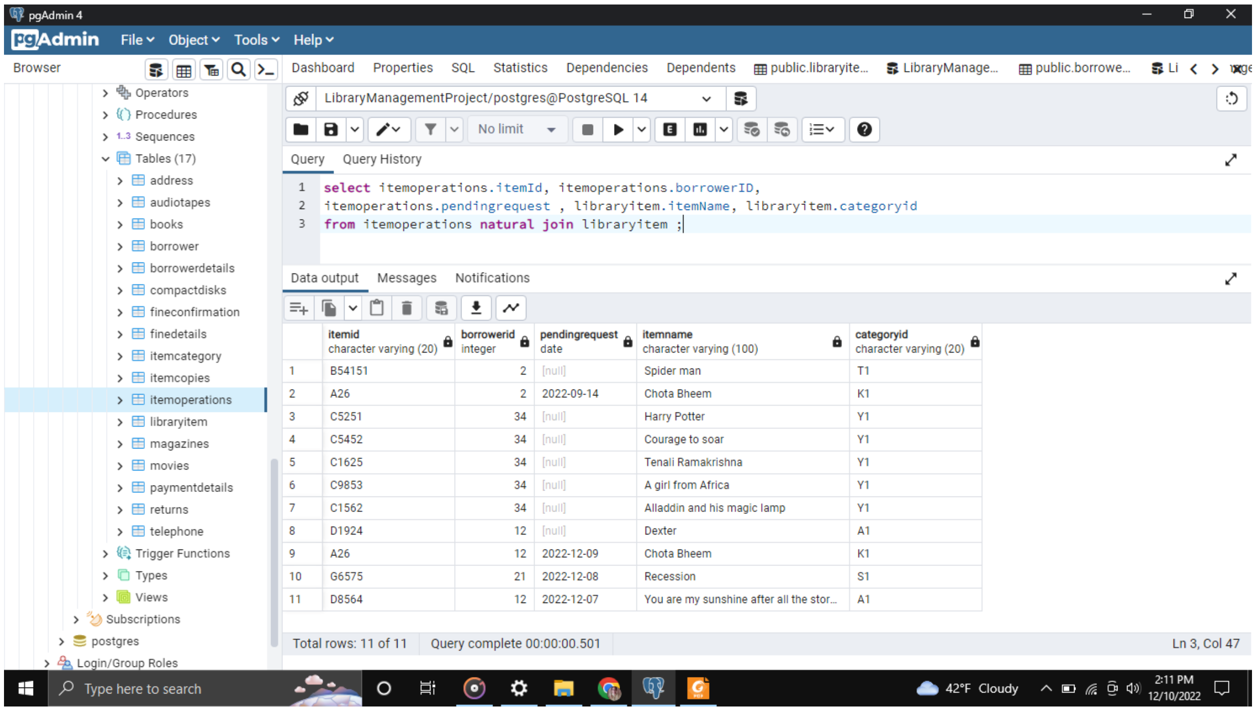
	itemname character varying (100)	finereason character varying (100)
1	Spider man	overdue
2	Super Man	overdue
3	Chota Bheem	overdue
4	Harry Potter	overdue
5	Courage to soar	overdue
6	You are my sunshine after all the stor...	overdue
7	Frankie and Gracy	overdue
8	Spider man	overdue

Total rows: 8 of 8 Query complete 00:00:03.517

6. List the details of the current pending requests in the system.

select itemoperations.itemId, itemoperations.borrowerID, itemoperations.pendingrequest , libraryitem.itemName, libraryitem.categoryid from itemoperations natural join libraryitem ;

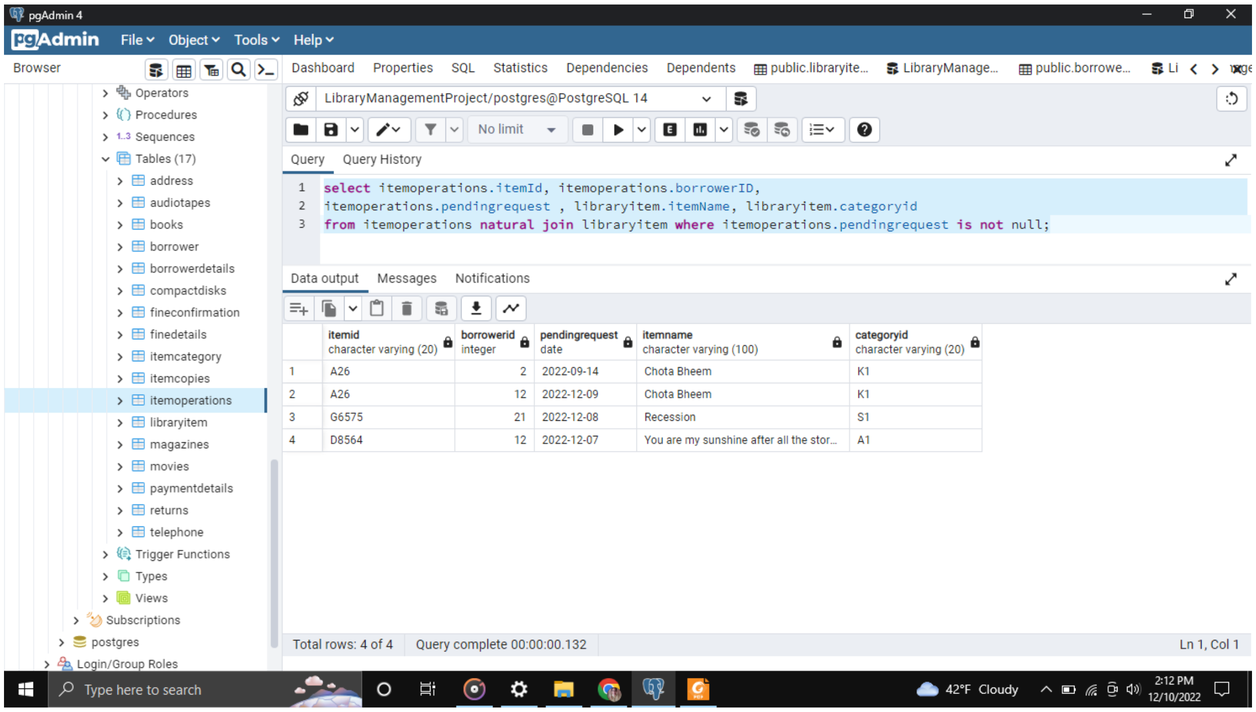
select itemoperations.itemId, itemoperations.borrowerID, itemoperations.pendingrequest , libraryitem.itemName, libraryitem.categoryid from itemoperations natural join libraryitem where itemoperations.pendingrequest is not null;



```sql
select itemoperations.itemId, itemoperations.borrowerID,
itemoperations.pendingrequest , libraryitem.itemName, libraryitem.categoryid
from itemoperations natural join libraryitem ;
```

	itemid character varying (20)	borrowerid integer	pendingrequest date	itemname character varying (100)	categoryid character varying (20)
1	B54151	2	[null]	Spider man	T1
2	A26	2	2022-09-14	Chota Bheem	K1
3	C5251	34	[null]	Harry Potter	Y1
4	C5452	34	[null]	Courage to soar	Y1
5	C1625	34	[null]	Tenali Ramakrishna	Y1
6	C9853	34	[null]	A girl from Africa	Y1
7	C1562	34	[null]	Alladdin and his magic lamp	Y1
8	D1924	12	[null]	Dexter	A1
9	A26	12	2022-12-09	Chota Bheem	K1
10	G6575	21	2022-12-08	Recession	S1
11	D8564	12	2022-12-07	You are my sunshine after all the stor...	A1

Total rows: 11 of 11 Query complete 00:00:00.501 Ln 3, Col 47



```sql
select itemoperations.itemId, itemoperations.borrowerID,
itemoperations.pendingrequest , libraryitem.itemName, libraryitem.categoryid
from itemoperations natural join libraryitem where itemoperations.pendingrequest is not null;
```

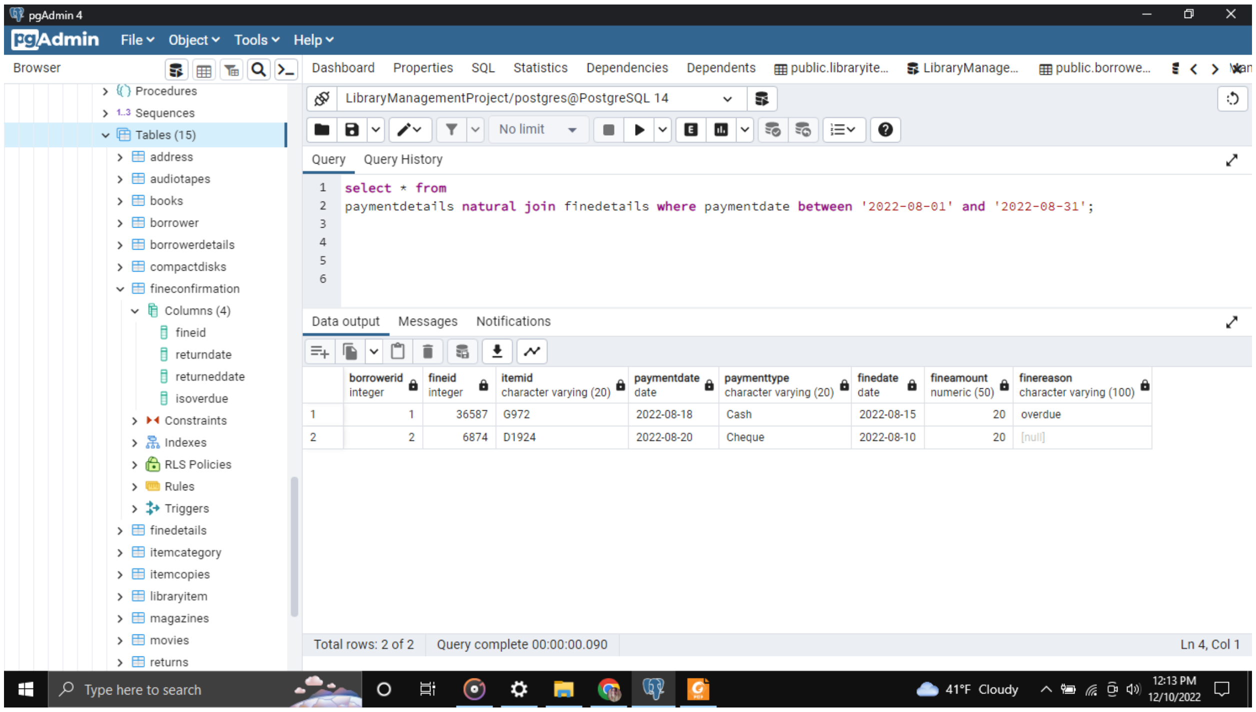
	itemid character varying (20)	borrowerid integer	pendingrequest date	itemname character varying (100)	categoryid character varying (20)
1	A26	2	2022-09-14	Chota Bheem	K1
2	A26	12	2022-12-09	Chota Bheem	K1
3	G6575	21	2022-12-08	Recession	S1
4	D8564	12	2022-12-07	You are my sunshine after all the stor...	A1

Total rows: 4 of 4 Query complete 00:00:00.132 Ln 1, Col 1

7. Identify the total fine revenue to the library between August 1, 2022 to August 31, 2022.

select Sum(finedetails.fineamount) from
paymentdetails natural join finedetails where paymentdate between '2022-08-01' and '2022-08-31';

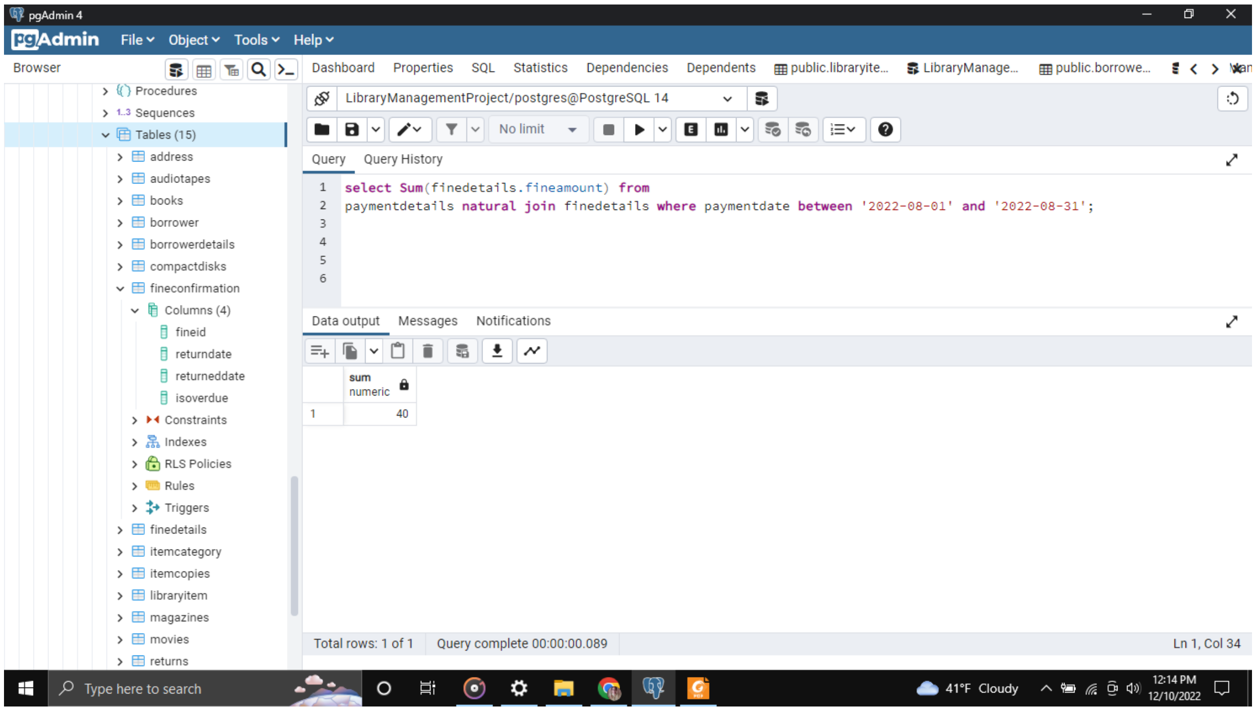
*select * from*
paymentdetails natural join finedetails where paymentdate between '2022-08-01' and '2022-08-31';



```sql
select * from
paymentdetails natural join finedetails where paymentdate between '2022-08-01' and '2022-08-31';
```

	borrowerid integer	fineid integer	itemid character varying (20)	paymentdate date	paymenttype character varying (20)	finedate date	fineamount numeric (50)	finereason character varying (100)
1	1	36587	G972	2022-08-18	Cash	2022-08-15	20	overdue
2	2	6874	D1924	2022-08-20	Cheque	2022-08-10	20	[null]

Total rows: 2 of 2 Query complete 00:00:00.090


```
select Sum(finedetails.fineamount) from
paymentdetails natural join finedetails where paymentdate between '2022-08-01' and '2022-08-31';
```

```
sum
numeric
1    40
```

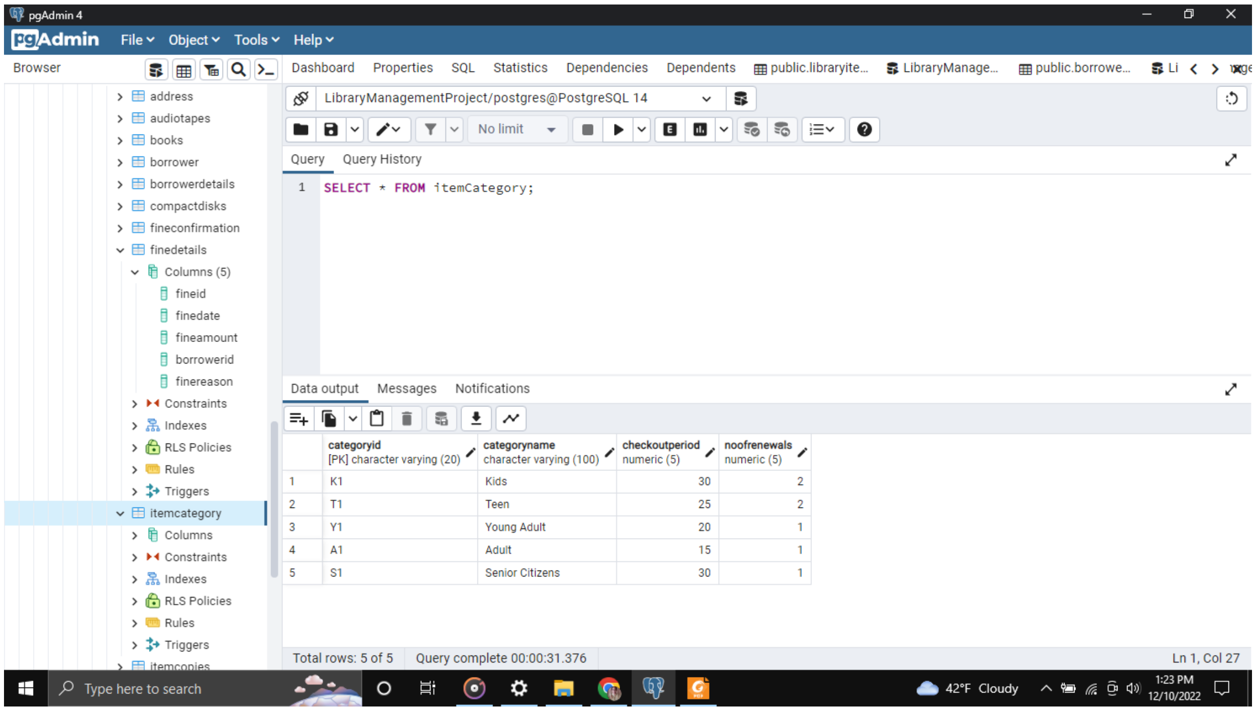
Total rows: 1 of 1   Query complete 00:00:00.089   Ln 1, Col 34

8. List the details of the checkout periods and the numbers of renewals for all the categories of library items.

The following query returns the checkout periods and the number of renewals an item belongs to each category can have.

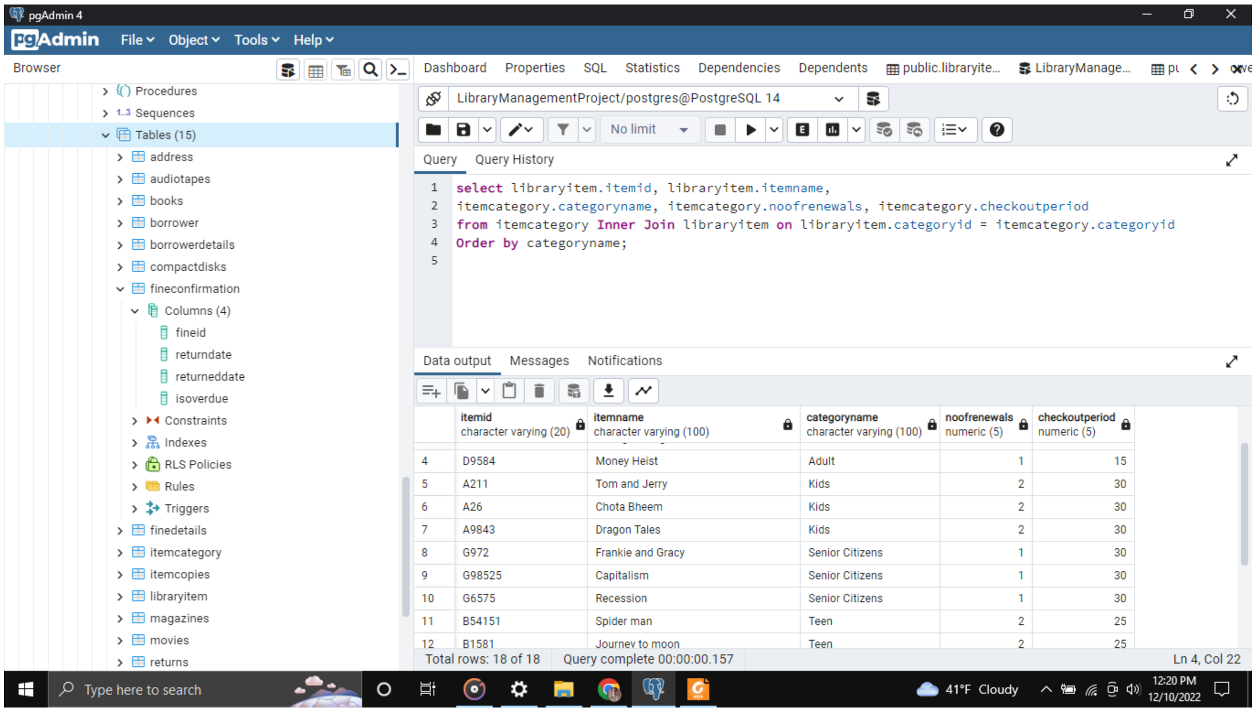
select libraryitem.itemid, libraryitem.itemname,
itemcategory.categoryname, itemcategory.noofrenewals,
itemcategory.checkoutperiod
from itemcategory Inner Join libraryitem on libraryitem.categoryid =
itemcategory.categoryid
Order by categoryname;

*Select * From itemcategory;*


```
SELECT * FROM itemCategory;
```

	categoryid [PK] character varying (20)	categoryname character varying (100)	checkoutperiod numeric (5)	noofrenewals numeric (5)
1	K1	Kids	30	2
2	T1	Teen	25	2
3	Y1	Young Adult	20	1
4	A1	Adult	15	1
5	S1	Senior Citizens	30	1

Total rows: 5 of 5 Query complete 00:00:31.376


```
select libraryitem.itemid, libraryitem.itemname,
itemcategory.categoryname, itemcategory.noofrenewals, itemcategory.checkoutperiod
from itemcategory Inner Join libraryitem on libraryitem.categoryid = itemcategory.categoryid
Order by categoryname;
```

	itemid character varying (20)	itemname character varying (100)	categoryname character varying (100)	noofrenewals numeric (5)	checkoutperiod numeric (5)
4	D9584	Money Heist	Adult	1	15
5	A211	Tom and Jerry	Kids	2	30
6	A26	Chota Bheem	Kids	2	30
7	A9843	Dragon Tales	Kids	2	30
8	G972	Frankie and Gracy	Senior Citizens	1	30
9	G98525	Capitalism	Senior Citizens	1	30
10	G6575	Recession	Senior Citizens	1	30
11	B54151	Spider man	Teen	2	25
12	B1581	Journey to moon	Teen	2	25

Total rows: 18 of 18 Query complete 00:00:00.157

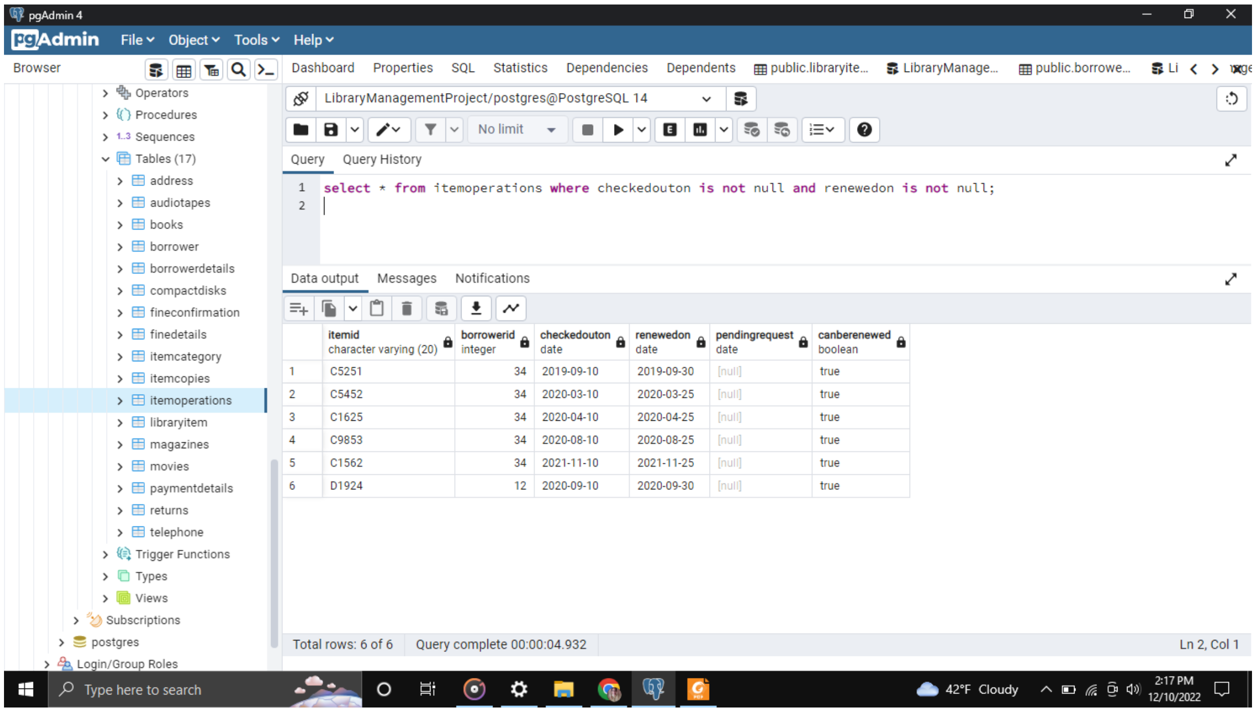
9. List the total number of library items that are checked out and renewed by a patron.

Finding out the total number of libraryites that were checked out and renewed by a borrower with borrowerId 12 and 34.

*select * from itemoperations where checkedouton is not null and renewedon is not null;*

select Count(itemid) from itemoperations
where checkedouton is not null and renewedon is not null and borrowerid = 12;

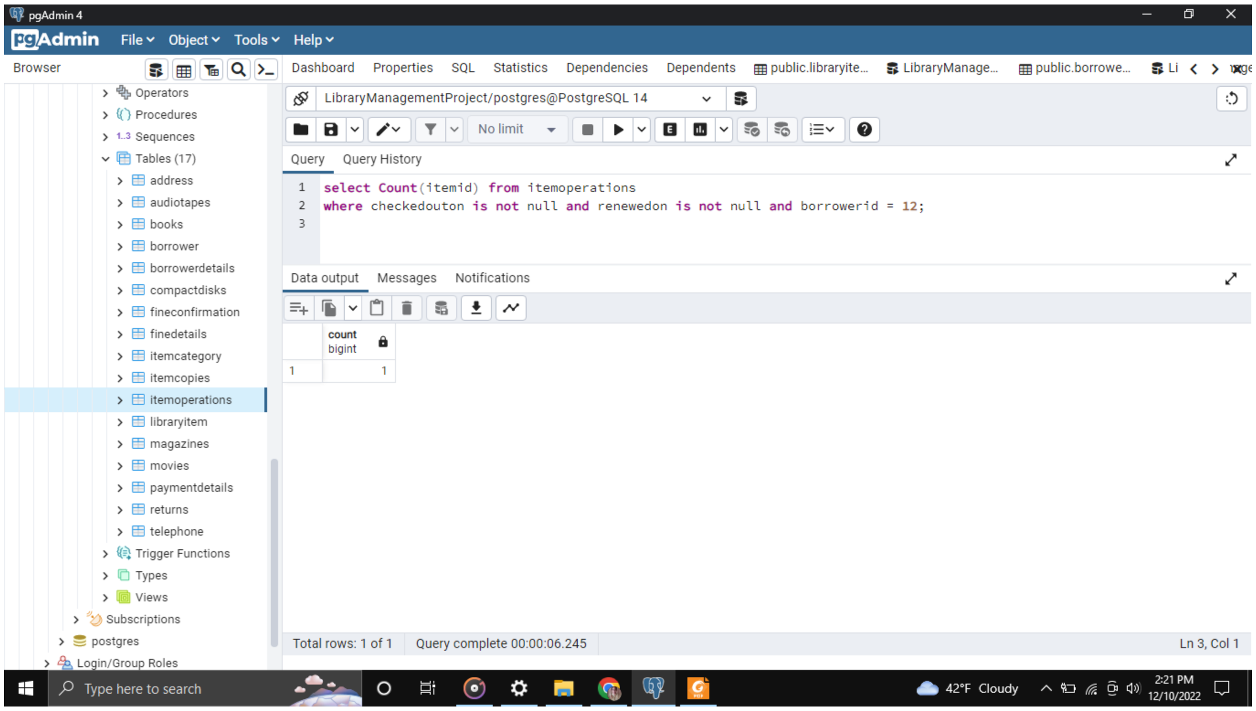
select Count(itemid) from itemoperations
where checkedouton is not null and renewedon is not null and borrowerid = 34;



```
select * from itemoperations where checkedouton is not null and renewedon is not null;
```

	itemid character varying (20)	borrowerid integer	checkedouton date	renewedon date	pendingrequest date	canberenewed boolean
1	C5251	34	2019-09-10	2019-09-30	[null]	true
2	C5452	34	2020-03-10	2020-03-25	[null]	true
3	C1625	34	2020-04-10	2020-04-25	[null]	true
4	C9853	34	2020-08-10	2020-08-25	[null]	true
5	C1562	34	2021-11-10	2021-11-25	[null]	true
6	D1924	12	2020-09-10	2020-09-30	[null]	true

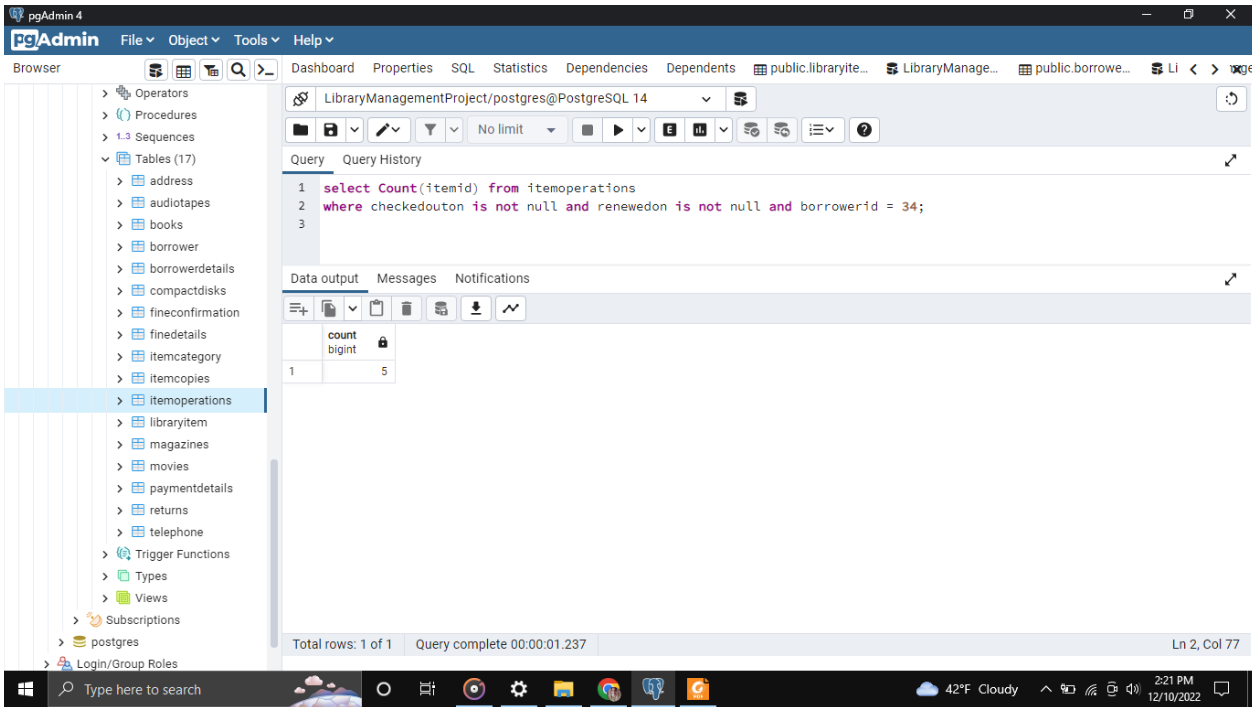
Total rows: 6 of 6 Query complete 00:00:04.932 Ln 2, Col 1



```
select Count(itemid) from itemoperations
where checkedouton is not null and renewedon is not null and borrowerid = 12;
```

```
  count
  bigint
1      1
```

Total rows: 1 of 1 Query complete 00:00:06.245



```
select Count(itemid) from itemoperations
where checkedouton is not null and renewedon is not null and borrowerid = 34;
```

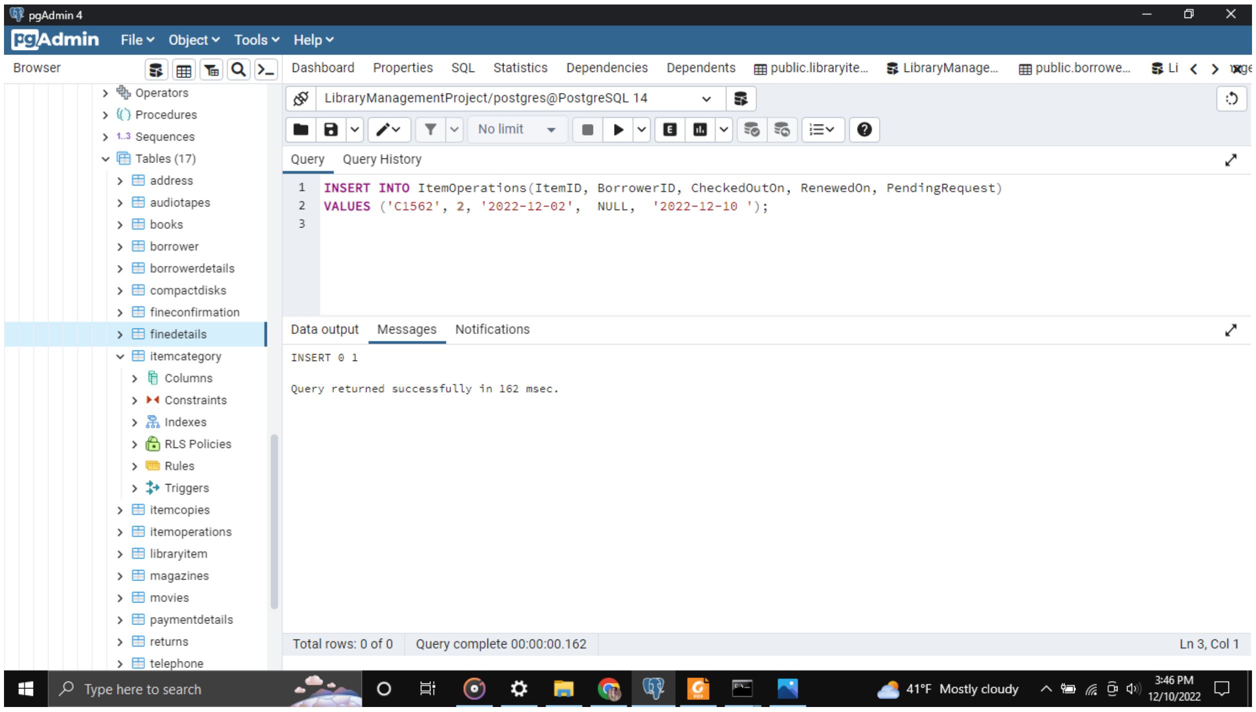
```
  count
  bigint
1      5
```

Total rows: 1 of 1 Query complete 00:00:01.237

10. Insert a new pending request made by a patron.

INSERT INTO ItemOperations(ItemID, BorrowerID, CheckedOutOn, RenewedOn, PendingRequest)
VALUES ('C9853', 2, '2022-12-02', NULL, '2022-12-10 ');

Borrower with ID 2 placed a pending request on 2022-12-10 for item id C9853 which was checked out by another borrower on 2022-12-02.



```
INSERT INTO ItemOperations(ItemID, BorrowerID, CheckedOutOn, RenewedOn, PendingRequest)
VALUES ('C1562', 2, '2022-12-02',  NULL,  '2022-12-10 ');
```

```
INSERT 0 1

Query returned successfully in 162 msec.
```

Total rows: 0 of 0   Query complete 00:00:00.162   Ln 3, Col 1