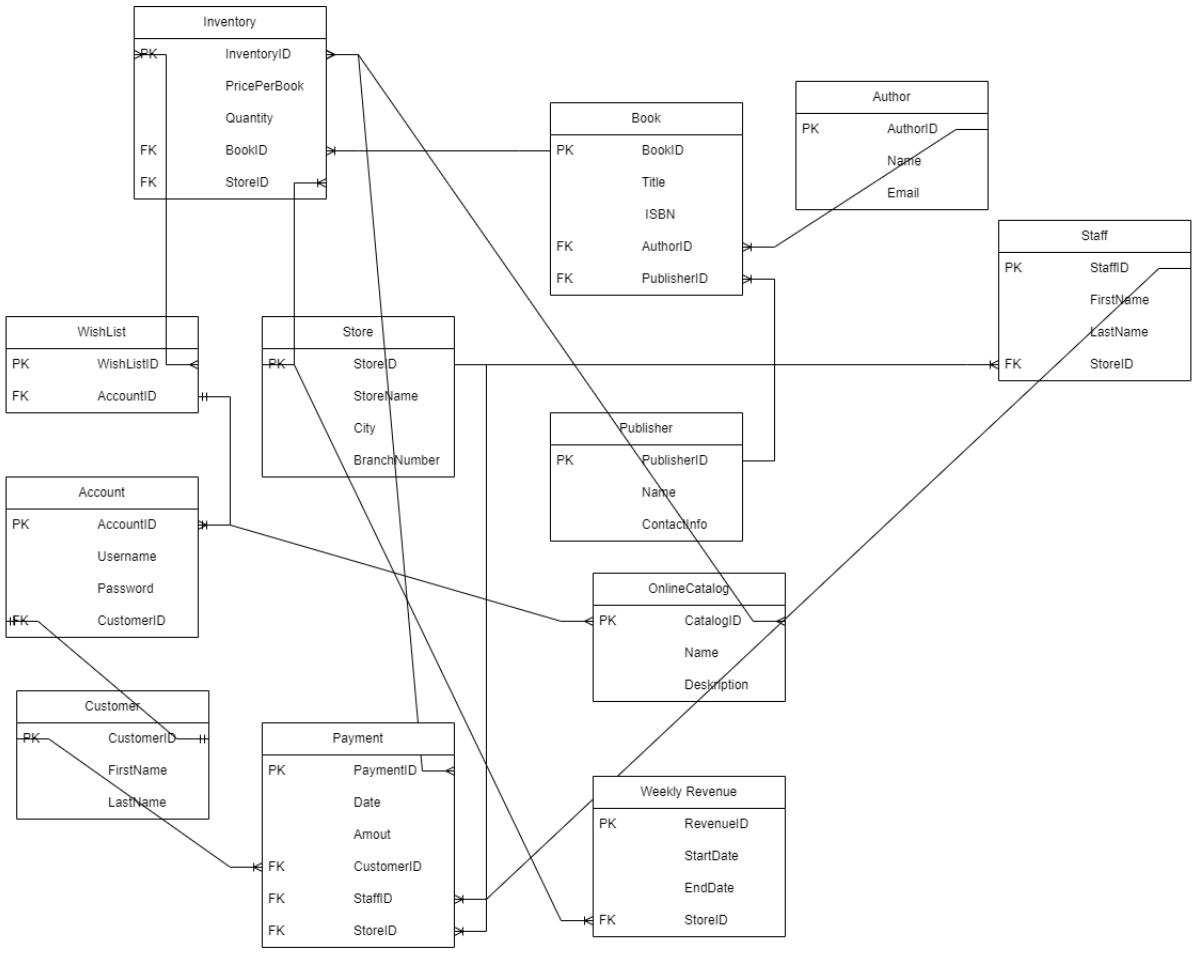
Nibroos Aurore Majiid Haryanto [22/494882/TK/54329]

Report of Expanding GRB Database

ERD Design



Establishing Table from the ERD

1. Book

BookID (PK)	Title	ISBN	AuthorID (FK)	PublisherID (FK)

2. Author

AuthorID (PK)	Name	Email

3. Publisher

PublisherID (PK)	Name	ContactInfo

4. Store

StoreID (PK)	StoreName	City	BranchNumber

5. Inventory

InventoryID (PK)	PricePerBook	Quantity	BookID (FK)	StoreID (FK)

6. Customer

CustomerID (PK)	FirstName	LastName

7. Account

AccountID (PK)	Username	Password	CustomerID (FK)

8. Payment

PaymentID (PK)	Date	Amount	CustomerID (FK)	StaffID (FK)	StoreID (FK)

9. WishList

WishListID (PK)	AccountID (FK)

10. OnlineCatalog

CatalogID (PK)	Name	Description

11. Staff

StaffID (PK)	FirstName	LastName	StoreID (FK)

12. WeeklyRevenue ()

RevenueID (PK)	StartDate	EndDate	StoreID (FK)

13. Inventory_WishList

WishListID (FK)	InventoryID (FK)

14. Inventory_OnlineCatalog

InventoryID (FK)	OnlineCatalogID (FK)

15. OnlineCatalog_Account

AccountID (FK)	OnlineCatalogID (FK)

16. Inventory_Payment

InventoryID (FK)	PaymentID (FK)

Generating Data Definition Language (DDL) from ERD using pdAdmin 4

```
-- This script was generated by the ERD tool in pgAdmin 4.
-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.
BEGIN;


CREATE TABLE IF NOT EXISTS public.account
(
    accountid serial NOT NULL,
    username character varying(50) COLLATE pg_catalog."default",
    passwordaccount character varying(20) COLLATE pg_catalog."default",
    customerid integer,
    CONSTRAINT account_pkey PRIMARY KEY (accountid)
);

CREATE TABLE IF NOT EXISTS public.customer
(
    customerid serial NOT NULL,
    firstname character varying(50) COLLATE pg_catalog."default",
    lastname character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT customer_pkey PRIMARY KEY (customerid)
);

CREATE TABLE IF NOT EXISTS public.payment
(
    paymentid serial NOT NULL,
    paymentdate date,
    amount bigint,
    customerid integer,
    staffid integer,
    storeid integer,
    CONSTRAINT payment_pkey PRIMARY KEY (paymentid)
);

CREATE TABLE IF NOT EXISTS public.staff
(
    staffid serial NOT NULL,
    firstname character varying(50) COLLATE pg_catalog."default",
    lastname character varying(20) COLLATE pg_catalog."default",
    storeid integer,
    CONSTRAINT staff_pkey PRIMARY KEY (staffid)
);

CREATE TABLE IF NOT EXISTS public.store
(
    storeid serial NOT NULL,
    storename character varying(50) COLLATE pg_catalog."default",
    city character varying(55) COLLATE pg_catalog."default",
    branchnumber integer,
    CONSTRAINT store_pkey PRIMARY KEY (storeid)
```

```
);

CREATE TABLE IF NOT EXISTS public.inventory
(
    inventoryid serial NOT NULL,
    priceperbook_rupiah numeric(10, 2),
    quantity integer,
    bookid integer,
    storeid integer,
    CONSTRAINT inventory_pkey PRIMARY KEY (inventoryid)
);

CREATE TABLE IF NOT EXISTS public.books
(
    bookid serial NOT NULL,
    title character varying(255) COLLATE pg_catalog."default",
    isbn character varying(20) COLLATE pg_catalog."default",
    authorid integer,
    publisherid integer,
    CONSTRAINT books_pkey PRIMARY KEY (bookid)
);

CREATE TABLE IF NOT EXISTS public.authors
(
    authorid serial NOT NULL,
    authorname character varying(55) COLLATE pg_catalog."default",
    email character varying(100) COLLATE pg_catalog."default",
    CONSTRAINT authors_pkey PRIMARY KEY (authorid)
);

CREATE TABLE IF NOT EXISTS public.publisher
(
    publisherid serial NOT NULL,
    publishername character varying(100) COLLATE pg_catalog."default",
    contactinfo character varying(100) COLLATE pg_catalog."default",
    CONSTRAINT publisher_pkey PRIMARY KEY (publisherid)
);

CREATE TABLE IF NOT EXISTS public.inventory_wishlist
(
    wishlistid integer,
    inventoryid integer
);

CREATE TABLE IF NOT EXISTS public.wishlist
(
    wishlistid serial NOT NULL,
    accountid integer,
    CONSTRAINT wishlist_pkey PRIMARY KEY (wishlistid)
);
```

```
CREATE TABLE IF NOT EXISTS public.inventory_onlinecatalog
(
    inventoryid integer,
    onlinecatalogid integer
);

CREATE TABLE IF NOT EXISTS public.onlinecatalog
(
    onlinecatalogid serial NOT NULL,
    namecatalog character varying(50) COLLATE pg_catalog."default",
    description character varying(300) COLLATE pg_catalog."default",
    CONSTRAINT onlinecatalog_pkey PRIMARY KEY (onlinecatalogid)
);

CREATE TABLE IF NOT EXISTS public.account_onlinecatalog
(
    accountid integer,
    onlinecatalogid integer
);

CREATE TABLE IF NOT EXISTS public.inventory_payment
(
    inventoryid integer,
    paymentid integer
);

CREATE TABLE IF NOT EXISTS public.weeklyrevenue
(
    revenueid serial NOT NULL,
    startdate date,
    enddate date,
    storeid integer,
    CONSTRAINT weeklyrevenue_pkey PRIMARY KEY (revenueid)
);

ALTER TABLE IF EXISTS public.account
    ADD CONSTRAINT account_customerid_fkey FOREIGN KEY (customerid)
    REFERENCES public.customer (customerid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.payment
    ADD CONSTRAINT payment_customerid_fkey FOREIGN KEY (customerid)
    REFERENCES public.customer (customerid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.payment
    ADD CONSTRAINT payment_staffid_fkey FOREIGN KEY (staffid)
```

```
    REFERENCES public.staff (staffid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.payment
    ADD CONSTRAINT payment_storeid_fkey FOREIGN KEY (storeid)
    REFERENCES public.store (storeid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.staff
    ADD CONSTRAINT staff_storeid_fkey FOREIGN KEY (storeid)
    REFERENCES public.store (storeid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory
    ADD CONSTRAINT inventory_bookid_fkey FOREIGN KEY (bookid)
    REFERENCES public.books (bookid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory
    ADD CONSTRAINT inventory_storeid_fkey FOREIGN KEY (storeid)
    REFERENCES public.store (storeid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.books
    ADD CONSTRAINT fk_author FOREIGN KEY (authorid)
    REFERENCES public.authors (authorid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.books
    ADD CONSTRAINT fk_publisher FOREIGN KEY (publisherid)
    REFERENCES public.publisher (publisherid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_wishlist
    ADD CONSTRAINT inventory_wishlist_inventoryid_fkey FOREIGN KEY (inventoryid)
    REFERENCES public.inventory (inventoryid) MATCH SIMPLE
    ON UPDATE NO ACTION
```

```
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_wishlist
    ADD CONSTRAINT inventory_wishlist_wishlistid_fkey FOREIGN KEY (wishlistid)
    REFERENCES public.wishlist (wishlistid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.wishlist
    ADD CONSTRAINT wishlist_accountid_fkey FOREIGN KEY (accountid)
    REFERENCES public.account (accountid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_onlinecatalog
    ADD CONSTRAINT inventory_onlinecatalog_onlinecatalogid_fkey FOREIGN KEY
(onlinecatalogid)
    REFERENCES public.onlinecatalog (onlinecatalogid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_onlinecatalog
    ADD CONSTRAINT inventory_onlinecatalog_wishlistid_fkey FOREIGN KEY (inventoryid)
    REFERENCES public.wishlist (wishlistid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.account_onlinecatalog
    ADD CONSTRAINT account_onlinecatalog_accountid_fkey FOREIGN KEY (accountid)
    REFERENCES public.account (accountid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.account_onlinecatalog
    ADD CONSTRAINT account_onlinecatalog_onlinecatalogid_fkey FOREIGN KEY
(onlinecatalogid)
    REFERENCES public.onlinecatalog (onlinecatalogid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_payment
    ADD CONSTRAINT inventory_payment_inventoryid_fkey FOREIGN KEY (inventoryid)
    REFERENCES public.inventory (inventoryid) MATCH SIMPLE
    ON UPDATE NO ACTION
```

```
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.inventory_payment
    ADD CONSTRAINT inventory_payment_paymentid_fkey FOREIGN KEY (paymentid)
    REFERENCES public.payment (paymentid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.weeklyrevenue
    ADD CONSTRAINT weeklyrevenue_storeid_fkey FOREIGN KEY (storeid)
    REFERENCES public.store (storeid) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;

END;
```

Populating Table and Scenario of TCL

1. Synchronizing 20 customers with 20 accounts

Query | Query History

```sql
BEGIN; -- Start a transaction

INSERT INTO Account (UserName, PasswordAccount, CustomerID)
VALUES
    ('DoeJohn', 'password1', 1),
    ('SmithJane', 'password2', 2),
    ('JohnMichael', 'password3', 3),
    ('LiamEmmily', 'password4', 8),
    ('BrownDan', 'password5', 5),
    ('JoneSarah', 'password6', 6),
    ('DavisCris', 'password7', 7),
    ('AmandaMill', 'password8', 4),
    ('MatTaylor', 'password9', 9),
    ('WilJecc', 'password10', 10),
    ('MooreDave', 'password11', 11),
    ('TomJames', 'password20', 20),
    ('WhiteJen', 'password13', 13),
    ('MartiniBert', 'password14', 14),
    ('JackMel', 'password15', 15),
    ('HarrWill', 'password16', 16),
    ('ThompStep', 'password17', 17),
    ('GarcSep', 'password18', 18),
    ('AureMartin', 'password19', 19),
    ('AshleyAnderson', 'password12', 12);

COMMIT; -- Commit the transaction
```

2. Adding 20 instances into Publisher Table

```sql
BEGIN;

INSERT INTO Publisher (PublisherName, ContactInfo)
VALUES
    ('Pearson Publishers', 'john@pearson.com'),
    ('Random House', 'jane@randomhouse.com'),
    ('Penguin Books', 'michael@penguinbooks.com'),
    ('HarperCollins', 'emily@harpercollins.com'),
    ('Macmillan Publishers', 'daniel@macmillan.com'),
    ('Hachette Book Group', 'sarah@hachettebooks.com'),
    ('Scholastic Corporation', 'christopher@scholastic.com'),
    ('Simon & Schuster', 'amanda@simonandschuster.com'),
    ('Houghton Mifflin Harcourt', 'matthew@hmhco.com'),
    ('Oxford University Press', 'jessica@oup.com'),
    ('Cambridge University Press', 'david@cup.org'),
    ('Wiley', 'ashley@wiley.com'),
    ('Elsevier', 'james@elsevier.com'),
    ('Harvard University Press', 'jennifer@harvardpress.com'),
    ('University of Chicago Press', 'robert@uchicago.edu'),
    ('Bloomsbury Publishing', 'melissa@bloomsbury.com'),
    ('Hogan Lovells', 'william@hoganlovells.com'),
    ('LexisNexis', 'stephanie@lexisnexis.com'),
    ('Wolters Kluwer', 'joseph@wolterskluwer.com'),
    ('Springer Nature', 'laura@springernature.com');

COMMIT;
```

3. Adding 20 instances into Authors Table

```sql
BEGIN; -- Start a transaction

INSERT INTO Authors (AuthorName, Email)
VALUES
    ('Jane Austen', 'jane.austen@gmail.com'),
    ('Mark Twain', 'mark.twain@gmail.com'),
    ('Charles Dickens', 'charles.dickens@gmail.com'),
    ('J.K. Rowling', 'jk.rowling@ugm.com'),
    ('Ernest Hemingway', 'ernest.hemingway@itb.com'),
    ('F. Scott Fitzgerald', 'f.scott.fitzgerald@dummy.com'),
    ('George Orwell', 'george.orwell@dummy.com'),
    ('Virginia Woolf', 'virginia.woolf@dummy.com'),
    ('Leo Tolstoy', 'leo.tolstoy@dummy.com'),
    ('Gabriel Garcia Marquez', 'gabriel.garcia.marquez@dummy.com'),
    ('Agatha Christie', 'agatha.christie@dummy.com'),
    ('J.R.R. Tolkien', 'jrr.tolkien@dummy.com'),
    ('William Faulkner', 'william.faulkner@dummy.com'),
    ('Herman Melville', 'herman.melville@dummy.com'),
    ('Lewis Carroll', 'lewis.carroll@dummy.com'),
    ('James Joyce', 'james.joyce@dummy.com'),
    ('Franz Kafka', 'franz.kafka@dummy.com'),
    ('Marcel Proust', 'marcel.proust@dummy.com'),
    ('Harper Lee', 'harper.lee@dummy.com'),
    ('Toni Morrison', 'toni.morrison@dummy.com');

COMMIT; -- Commit the transaction
```

4. Inserting 20 instances of books with the corresponding author and publisher

```sql
BEGIN; -- Start a transaction

-- Insert 20 instances of books with titles, ISBN, AuthorID, and PublisherID
INSERT INTO Books (Title, ISBN, AuthorID, PublisherID)
VALUES
    ('Pride and Prejudice', '978-3-16-148410-0', 1, 3),
    ('Emma', '978-3-16-148411-7', 1, 3),
    ('Sense and Sensibility', '978-3-16-148412-4', 1, 3),
    ('Harry Potter and the Deathly Hallows', '978-3-16-148413-1', 4, 16),
    ('Harry Potter and the Chamber of Secrets', '978-3-16-148414-8', 4, 16),
    ('Harry Potter and the Prisoner of Azkaban', '978-3-16-148415-5', 4, 16),
    ('1984', '978-3-16-148416-2', 7, 3),
    ('Animal Farm', '978-3-16-148417-9', 7, 3),
    ('Homage to Catalonia', '978-3-16-148418-6', 7, 9),
    ('Mrs Dalloway', '978-3-16-148419-3', 8, 4),
    ('To the Lighthouse', '978-3-16-148420-9', 8, 4),
    ('Orlando', '978-3-16-148421-6', 8, 4),
    ('The Hobbit', '978-3-16-148422-3', 12, 3),
    ('The Lord of the Rings: The Fellowship of the Ring', '978-3-16-148423-0', 12, 3),
    ('The Lord of the Rings: The Two Towers', '978-3-16-148424-7', 12, 3),
    ('The Metamorphosis', '978-3-16-148425-4', 17, 9),
    ('The Trial', '978-3-16-148426-1', 17, 9),
    ('Alice: Adventures in Wonderland', '978-3-16-148427-8', 15, 3),
    ('Alice: Through the Looking-Glass', '978-3-16-148428-5', 15, 3),
    ('The Hunting of the Snark', '978-3-16-148429-2', 15, 4);

COMMIT; -- Commit the transaction
```

5. Adding 20 instances into Store Table

```sql
BEGIN; -- Start a transaction

-- Insert 5 instances of stores with StoreName, City, and BranchNumber
INSERT INTO Store (StoreName, City, BranchNumber)
VALUES
    ('Good Reading Book Store cabang Manahan', 'Solo', 1),
    ('Good Reading Book Store cabang Kridosono', 'Yogyakarta', 2),
    ('Good Reading Book Store cabang Klebengan', 'Sleman', 3),
    ('Good Reading Book Store cabang Alkid', 'Yogyakarta', 4),
    ('Good Reading Book Store cabang Pantai Congot', 'Kulon Progo', 5);

COMMIT; -- Commit the transaction
```

One of the task's specifications is "Populating some dummy data; at least 20 records for each table". Though, the GRB just about to expand their business and I think opening 20 stores at once is more than risky given that Indonesia has a small amount of literacy index. As the database engineer, I would present my idea to just open 5 store at a time to the stockholders so GRB could survive in Indonesia. Therefore, in this table I only made 5 entries.

6. Adding 20 instances into Inventory Table

```sql
BEGIN; -- Start a transaction

-- Insert 20 instances into Inventory with random Quantity, BookID, and StoreID
INSERT INTO Inventory (PricePerBook_rupiah, Quantity, BookID, StoreID)
SELECT
    FLOOR(RANDOM() * 300000) + 100000 AS PricePerBook, -- Random price
    FLOOR(RANDOM() * 50) + 1 AS Quantity, -- Random quantity
    FLOOR(RANDOM() * 20) + 1 AS BookID, -- Random BookID between 1 and 20
    FLOOR(RANDOM() * 5) + 1 AS StoreID -- Random StoreID between 1 and 5
FROM generate_series(1, 20); -- Generate 20 rows

COMMIT; -- Commit the transaction
```

7. Adding 20 instances into OnlineCatalog Table

```sql
BEGIN; -- Start a transaction

INSERT INTO OnlineCatalog (NameCatalog, Description)
VALUES
    ('Best Seller Catalog', 'Discover our best-selling books that have captured the hearts of many readers. Explore a variety
    ('Classical Catalog', 'Dive into the timeless classics that have stood the test of time. From Shakespeare to Jane Austen,
    ('Hits', 'Explore the latest hits in the literary world. From thrilling mysteries to heartwarming romances, find the books
    ('TiktokBook', 'Discover the books trending on TikTok! From viral book recommendations to bookish challenges, find your ne
    ('Gen-Z Banget', 'For the Gen-Z readers out there! Explore books that resonate with your generation, from YA fiction to cc
    ('Summer Reads', 'Find the perfect reads for your summer escape. Whether you are lounging on the beach or relaxing in your
    ('Sci-Fi & Fantasy', 'Embark on epic adventures in distant galaxies or magical realms. Explore the limitless possibilities
    ('Mystery & Thriller', 'Get your heart racing with gripping mysteries and pulse-pounding thrillers. Prepare for twists anc
    ('Romance', 'Indulge in love stories that will sweep you off your feet. From sweet romances to passionate encounters, finc
    ('Non-Fiction', 'Expand your horizons with thought-provoking non-fiction books. From memoirs to self-help guides, explore
    ('Historical Fiction', 'Step back in time with captivating historical fiction novels. Immerse yourself in the rich tapestr
    ('Biographies', 'Discover the remarkable lives of extraordinary individuals. From world leaders to cultural icons, explore
    ('Self-Help & Personal Development', 'Invest in yourself with self-help and personal development books. Unlock your full p
    ('Cookbooks', 'Satisfy your culinary cravings with mouthwatering recipes from around the world. From gourmet delights to c
    ('Poetry', 'Experience the beauty of language through poetry. From classic poets to contemporary voices, explore the power
    ('Art & Photography', 'Immerse yourself in the world of art and photography. From breathtaking landscapes to stunning port
    ('Travel', 'Embark on a journey of discovery with travel books that transport you to faraway destinations. From guidebooks
    ('Science & Nature', 'Unlock the mysteries of the universe with science and nature books. From the depths of the ocean to
    ('Business & Finance', 'Take charge of your financial future with business and finance books. From investing strategies to
    ('Health & Wellness', 'Prioritize your well-being with health and wellness books. From fitness regimes to mental health re

COMMIT; -- Commit
```

8. Adding 20 instances into Staff Table

Query | Query History

```sql
BEGIN;
INSERT INTO Staff (FirstName, LastName, StoreID)
VALUES
    ('Nadia', 'Smith', FLOOR(RANDOM() * 5) + 1),
    ('Rizky', 'Johnson', FLOOR(RANDOM() * 5) + 1),
    ('Dewi', 'Williams', FLOOR(RANDOM() * 5) + 1),
    ('Ahmad', 'Brown', FLOOR(RANDOM() * 5) + 1),
    ('Siti', 'Jones', FLOOR(RANDOM() * 5) + 1),
    ('Fajar', 'Taylor', FLOOR(RANDOM() * 5) + 1),
    ('Laras', 'Wilson', FLOOR(RANDOM() * 5) + 1),
    ('Eko', 'Davis', FLOOR(RANDOM() * 5) + 1),
    ('Dian', 'White', FLOOR(RANDOM() * 5) + 1),
    ('Andi', 'Clark', FLOOR(RANDOM() * 5) + 1),
    ('Budi', 'Hall', FLOOR(RANDOM() * 5) + 1),
    ('Tia', 'Lewis', FLOOR(RANDOM() * 5) + 1),
    ('Arief', 'Moore', FLOOR(RANDOM() * 5) + 1),
    ('Ratna', 'Walker', FLOOR(RANDOM() * 5) + 1),
    ('Rudi', 'King', FLOOR(RANDOM() * 5) + 1),
    ('Sinta', 'Green', FLOOR(RANDOM() * 5) + 1),
    ('Adi', 'Hill', FLOOR(RANDOM() * 5) + 1),
    ('Dini', 'Scott', FLOOR(RANDOM() * 5) + 1),
    ('Wati', 'Adams', FLOOR(RANDOM() * 5) + 1),
    ('Rina', 'Cook', FLOOR(RANDOM() * 5) + 1);
COMMIT;
```

9. Adding 20 instances into WishList Table

Query | Query History

```sql
BEGIN;
INSERT INTO WishList (AccountID)
VALUES
    (1),
    (2),
    (3),
    (4),
    (5),
    (6),
    (7),
    (8),
    (9),
    (10),
    (11),
    (12),
    (13),
    (14),
    (15),
    (16),
    (17),
    (18),
    (19),
    (20);
COMMIT;
```

10. Adding several instances into Payment Table
    a. Transaction

Query | Query History

```sql
BEGIN;

-- Populate Payment Table
INSERT INTO payment (PaymentDate, Amount, CustomerID, StaffID, StoreID)
SELECT
    CURRENT_DATE AS PaymentDate,
    (SELECT 2 * priceperbook_rupiah FROM inventory WHERE inventoryid = 1) AS Amount, --note: I bought two book
    7 AS CustomerID,
    51 AS StaffID,
    1 AS StoreID; -- Adjust StaffID and StoreID as needed

-- Rollback the transaction if there's an error
SAVEPOINT payment_saved;

-- Check if the payment amount is calculated correctly
SELECT * FROM payment WHERE PaymentID = (SELECT (PaymentID) FROM payment);

-- Update the quantity in inventory
UPDATE inventory
SET quantity = quantity - 2 -- Assuming I bought two books
WHERE inventoryid = 1;

-- If everything is correct, commit the transaction
COMMIT;
```

As an attempt to populate the payment table I used TCL to establish the transaction. Firstly, this scenario is still flawed so I just used assumptions. For instance, if for one payment could only happen in one particular store, therefore before I initiate the transaction, I checked the staff table and inventory table to make sure that the *staffid* and *inventoryid* are located in the same store. Therefore, I got these following attributes *staffid* = 51 and *inventoryid* = 1 (*inventoryid* is used to calculate amount in payment table). Those attributes are in the same store, which is *storeid* = 1.

    b. Another Payment Transaction and Inventory Status

- Before transaction:

	inventoryid [PK] integer	priceperbook_rupiah numeric (10,2)	quantity integer	bookid integer	storeid integer
1	2	336874.00	36	4	4

- Transaction and Assumptions:

  I want to make a scenario to buy from *inventoryid* = 2 and it's located where *storeid* = 4 and *staffid* = 44. In this scheme I bought three books from *inventoryid* =2

```sql
BEGIN TRANSACTION;

-- Populate Payment Table
INSERT INTO payment (PaymentDate, Amount, CustomerID, StaffID, StoreID)
SELECT
    CURRENT_DATE AS PaymentDate,
    (SELECT 3 * priceperbook_rupiah FROM inventory WHERE inventoryid = 2) AS Amount,
    5 AS CustomerID,
    44 AS StaffID,
    4 AS StoreID; -- Adjust StaffID and StoreID as needed

-- Rollback the transaction if there's an error
SAVEPOINT payment_saved;

-- Check if the payment amount is calculated correctly
SELECT * FROM payment WHERE PaymentID = (SELECT MAX(PaymentID) FROM payment);

-- Update the quantity in inventory
UPDATE inventory
SET quantity = quantity - 3 -- Assuming you bought three books
WHERE inventoryid = 2;

-- If everything is correct, commit the transaction
COMMIT;

-- If there's an error, rollback to the savepoint and terminate the transaction
ROLLBACK TO payment_saved;
```

- After Transcation

	inventoryid [PK] integer	priceperbook_rupiah numeric (10,2)	quantity integer	bookid integer	storeid integer
1	2	336874.00	33	4	4

The quantity from the inventory table is being updated.

c. Payment Transaction

With the same method fill the Payment table up to 20 entries.

11. Adding several instances into Inventory_Payment Table

```
INSERT INTO Inventory_Payment (InventoryID, PaymentID)
VALUES (1, 6);

INSERT INTO Inventory_Payment (InventoryID, PaymentID)
VALUES (2, 8);
```

In this example, I filled the Inventory_Payment table based on the transaction that has been made from the previous point.

Query:

	inventoryid integer	paymentid integer
1	1	6
2	2	8

12. Adding 20 instances WeeklyRevenue

```
BEGIN;
INSERT INTO weeklyRevenue (StartDate, EndDate, StoreID)
VALUES
    (CURRENT_DATE - INTERVAL '13 days', CURRENT_DATE - INTERVAL '6 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '20 days', CURRENT_DATE - INTERVAL '13 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '27 days', CURRENT_DATE - INTERVAL '20 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '34 days', CURRENT_DATE - INTERVAL '27 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '41 days', CURRENT_DATE - INTERVAL '34 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '48 days', CURRENT_DATE - INTERVAL '41 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '55 days', CURRENT_DATE - INTERVAL '48 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '62 days', CURRENT_DATE - INTERVAL '55 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '69 days', CURRENT_DATE - INTERVAL '62 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '76 days', CURRENT_DATE - INTERVAL '69 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '83 days', CURRENT_DATE - INTERVAL '76 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '90 days', CURRENT_DATE - INTERVAL '83 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '97 days', CURRENT_DATE - INTERVAL '90 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '104 days', CURRENT_DATE - INTERVAL '97 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '111 days', CURRENT_DATE - INTERVAL '104 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '118 days', CURRENT_DATE - INTERVAL '111 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '125 days', CURRENT_DATE - INTERVAL '118 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '132 days', CURRENT_DATE - INTERVAL '125 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '139 days', CURRENT_DATE - INTERVAL '132 days', FLOOR(RANDOM() * 5) + 1),
    (CURRENT_DATE - INTERVAL '146 days', CURRENT_DATE - INTERVAL '139 days', FLOOR(RANDOM() * 5) + 1);
COMMIT;
```

13. Adding 20 instances Account_OnlineCatalog

```
BEGIN;
INSERT INTO Account_OnlineCatalog (AccountID, OnlineCatalogID)
VALUES
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1),
    (FLOOR(RANDOM() * 20) + 1, FLOOR(RANDOM() * 20) + 1);
COMMIT;
```

14. Adding several instances to the other tables

With the same method we could populate the remaining tables.

View

In this section, I would like to select * from *certain_table*, but not all the tables are included because it would be redundant. Therefore, I have shown only a few tables. In the image, there are 20 instances from the table, but not all the instances are included to keep the report short.

1. Table store

```
select * from store
```

	storeid [PK] integer	storename character varying (50)	city character varying (55)	branchnumber integer
1	1	Good Reading Book Store cabang Manahan	Solo	1
2	2	Good Reading Book Store cabang Kridosono	Yogyakarta	2
3	3	Good Reading Book Store cabang Klebengan	Sleman	3
4	4	Good Reading Book Store cabang Alkid	Yogyakarta	4
5	5	Good Reading Book Store cabang Pantai Congot	Kulon Progo	5

2. Table books

```
select * from books
```

	bookid [PK] integer	title character varying (255)	isbn character varying (20)	authorid integer	publisherid integer
2	2	Emma	978-3-16-148411-7	1	3
3	3	Sense and Sensibility	978-3-16-148412-4	1	3
4	4	Harry Potter and the Deathly Hallows	978-3-16-148413-1	4	16
5	5	Harry Potter and the Chamber of Secrets	978-3-16-148414-8	4	16
6	6	Harry Potter and the Prisoner of Azkaban	978-3-16-148415-5	4	16
7	7	1984	978-3-16-148416-2	7	3

3. Table author

```
select * from authors
```

	authorid [PK] integer	authorname character varying (55)	email character varying (100)
2	2	Mark Twain	mark.twain@gmail.com
3	3	Charles Dickens	charles.dickens@gmail.com
4	4	J.K. Rowling	jk.rowling@ugm.com
5	5	Ernest Hemingway	ernest.hemingway@itb.com
6	6	F. Scott Fitzgerald	f.scott.fitzgerald@dummy.com
7	7	George Orwell	george.orwell@dummy.com
8	8	Virginia Woolf	virginia.woolf@dummy.com

4. Table publisher

```
select * from publisher
```

	publisherid [PK] integer	publishername character varying (100)	contactinfo character varying (100)
1	1	Pearson Publishers	john@pearson.com
2	2	Random House	jane@randomhouse.com
3	3	Penguin Books	michael@penguinbooks.com
4	4	HarperCollins	emily@harpercollins.com
5	5	Macmillan Publishers	daniel@macmillan.com
6	6	Hachette Book Group	sarah@hachettebooks.com
7	7	Scholastic Corporation	christopher@scholastic.com
8	8	Simon & Schuster	amanda@simonandschuster.com
9	9	Houghton Mifflin Harcourt	matthew@hmhco.com

5. Table customer

```
Query  Query History
1  select * from customer
```

Data Output | Messages | Notifications

	customerid [PK] integer	firstname character varying (50)	lastname character varying (50)
3	3	Michael	Johnson
4	4	Emily	Williams
5	5	Daniel	Brown
6	6	Sarah	Jones
7	7	Christopher	Davis
8	8	Amanda	Miller
9	9	Matthew	Taylor
10	10	Jessica	Wilson
11	11	David	Moore
12	12	Ashley	Anderson

6. Table account

```
1  select * from account
```

Data Output | Messages | Notifications

	accountid [PK] integer	username character varying (50)	passwordaccount character varying (20)	customerid integer
1	1	DoeJohn	password1	1
2	2	SmithJane	password2	2
3	3	JohnMichael	password3	3
4	4	LiamEmmily	password4	8
5	5	BrownDan	password5	5
6	6	JoneSarah	password6	6
7	7	DavisCris	password7	7
8	8	AmandaMill	password8	4
9	9	MatTaylor	password9	9
10	10	WilJecc	password10	10
11	11	MooreDave	password11	11

7. Table staff

```
Query  Query History
1  select * from staff
```

Data Output | Messages | Notifications

	staffid [PK] integer	firstname character varying (50)	lastname character varying (20)	storeid integer
1	41	Nadia	Smith	1
2	42	Rizky	Johnson	1
3	43	Dewi	Williams	3
4	44	Ahmad	Brown	4
5	45	Siti	Jones	5
6	46	Fajar	Taylor	1
7	47	Laras	Wilson	2
8	48	Eko	Davis	5
9	49	Dian	White	2
10	50	Andi	Clark	4
11	51	Budi	Hall	1
12	52	Tia	Lewis	3
13	53	Arief	Moore	4

8. Table OnlineCatalog

```
1  select * from onlinecatalog
```

Data Output | Messages | Notifications

	onlinecatalogid [PK] integer	namecatalog character varying (50)	description character varying (300)
1	1	Best Seller Catalog	Discover our best-selling books that have captured the hearts of many readers. Explore a variety of genres
2	2	Classical Catalog	Dive into the timeless classics that have stood the test of time. From Shakespeare to Jane Austen, explor
3	3	Hits	Explore the latest hits in the literary world. From thrilling mysteries to heartwarming romances, find the bo
4	4	TiktokBook	Discover the books trending on TikTok! From viral book recommendations to bookish challenges, find you
5	5	Gen-Z Banget	For the Gen-Z readers out there! Explore books that resonate with your generation, from YA fiction to cont
6	6	Summer Reads	Find the perfect reads for your summer escape. Whether you are lounging on the beach or relaxing in your
7	7	Sci-Fi & Fantasy	Embark on epic adventures in distant galaxies or magical realms. Explore the limitless possibilities of scie
8	8	Mystery & Thriller	Get your heart racing with gripping mysteries and pulse-pounding thrillers. Prepare for twists and turns tha
9	9	Romance	Indulge in love stories that will sweep you off your feet. From sweet romances to passionate encounters, t
10	10	Non-Fiction	Expand your horizons with thought-provoking non-fiction books. From memoirs to self-help guides, explor
11	11	Historical Fiction	Step back in time with captivating historical fiction novels. Immerse yourself in the rich tapestry of the pa
12	12	Biographies	Discover the remarkable lives of extraordinary individuals. From world leaders to cultural icons, explore th
13	13	Self-Help & Personal Development	Invest in yourself with self-help and personal development books. Unlock your full potential and achieve y
14	14	Cookbooks	Satisfy your culinary cravings with mouthwatering recipes from around the world. From gourmet delights t
15	15	Poetry	Experience the beauty of language through poetry. From classic poets to contemporary voices, explore the

ERD Design from pgAdmin 4