Project Part 5: Physical Design and Data Population

Introduction:

Project Overview – The library management system detailed in this project will provide all the functionality and features a library might need to track, manage, and maintain their catalogue. It will contain details on media items such as Books, Magazines, and Digital Media. These items will have their own unique attributes as well as the attributes common on any media item. These items will be managed by the system and controlled by transactions. Transactions will be saved in the system with all their respective information, and they are initiated by clients. Clients will exist in the system with all their respective information as they're the driving users of the system.

Glossary -

- GUI – Graphical user interface
- ISBN – International Standard Book Number
- PostgreSQL – A relational database management system designed to efficiently manage structured data with complex relationships
- ACID – A set of principles that ensure database transactions are processed reliably (Atomicity, Concurrency, Isolation, Durability)
- ID (Identity) – A unique identifier assigned to entities such as books, users, or transactions in the database.
- FK – Foreign Key
- AWS Instance – A database environment created through AWS that runs independently and can be accessed remotely.
- DDL – Data Definition Language used to define, create, modify, and delete database structures.

Choosing your platform – AWS RDS + PostgreSQL:

We decided to host our database through Amazon Web Services. We decided to use AWS so everyone on our team can easily access the database from anywhere. The database itself is hosted through the AWS Relational Database Service as an "instance". The only information needed to access it is the domain of the database's endpoint and its master password. Normally permissions would be granted through user roles, but to keep our sign-in process simple we opted to use a master password.

For the database engine, we decided to use PostgreSQL. It is an open-source relational database management system. We decided to choose this since it is growing in popularity and was the most popular database system in 2024 according to the Stack Overflow developer survey. Additionally, it is also a database engine that none of our team is familiar with which made it an ideal choice as a learning experience. It also is a practical choice in combination with AWS since the RDS service supports PostgreSQL databases.

Creating the Database:

The database itself was created within Amazon Web Services as an RDS instance. Under the free plan, a single database instance of class “db.t4g.micro”, can be hosted. This database has 2 virtual CPU cores and 1 gigabyte of memory and runs on the PostgreSQL engine. It is configured to be publicly accessible while being secured with a master password. By knowing both the endpoint and master password, anyone on the group can access the database from anywhere.

Physical Schema (DDL Statements):

Below are the DDL statements used to create the tables within the database:

```
DO $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'membership_type_enum') THEN
    CREATE TYPE membership_type_enum AS ENUM ('Regular', 'Student', 'Senior Citizen', 'Other');
  END IF;
END $$;

DO $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'account_status_enum') THEN
    CREATE TYPE account_status_enum AS ENUM ('Active', 'Suspended', 'Inactive');
```

```
    END IF;
END $$;

DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'availability_status_enum')
THEN
        CREATE TYPE availability_status_enum AS ENUM ('Available', 'Unavailable');
    END IF;
END $$;

CREATE TABLE IF NOT EXISTS Client (
    client_id SERIAL PRIMARY KEY CHECK (client_id > 0),
    name VARCHAR(50) NOT NULL,
    membership_type membership_type_enum NOT NULL,
    account_status account_status_enum NOT NULL,
    email_address VARCHAR(50) UNIQUE NOT NULL,
    phone_number VARCHAR(15) UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS Media_Item (
    item_id SERIAL PRIMARY KEY CHECK (item_id > 0),
    availability_status availability_status_enum NOT NULL
);

CREATE TABLE IF NOT EXISTS Book (
```

```
    item_id INT PRIMARY KEY,
    FOREIGN KEY (item_id) REFERENCES Media_Item(item_id) ON DELETE CASCADE,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(100) NOT NULL,
    isbn VARCHAR(20) NOT NULL UNIQUE CHECK (isbn <> '0' AND isbn ~ '[1-9]'),
    genre VARCHAR(50),
    publication_year INT
);

CREATE TABLE IF NOT EXISTS Digital_Media (
    item_id INT PRIMARY KEY,
    FOREIGN KEY (item_id) REFERENCES Media_Item(item_id) ON DELETE CASCADE,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(100) NOT NULL,
    isbn VARCHAR(20) NOT NULL UNIQUE CHECK (isbn <> '0' AND isbn ~ '[1-9]'),
    genre VARCHAR(50),
    publication_year INT
);


CREATE TABLE IF NOT EXISTS Magazine (
    item_id INT PRIMARY KEY,
    FOREIGN KEY (item_id) REFERENCES Media_Item(item_id) ON DELETE CASCADE,
    title VARCHAR(100) NOT NULL,
    publication_date DATE NOT NULL,
    issue_number INT NOT NULL
```

```
);

CREATE TABLE IF NOT EXISTS Transaction (
   transaction_id SERIAL PRIMARY KEY CHECK (transaction_id >= 0),
   client_id INT NOT NULL,
   FOREIGN KEY (client_id) REFERENCES Client(client_id),
   item_id INT NOT NULL,
   FOREIGN KEY (item_id) REFERENCES Media_Item(item_id),
   date_borrowed TIMESTAMP NOT NULL,
   expected_return_date TIMESTAMP NOT NULL CHECK (expected_return_date > date_borrowed),
   returned_date TIMESTAMP CHECK (returned_date IS NULL OR returned_date > date_borrowed)
);
```

Data Population:

To populate the database, I wrote a Python script that connects to our PostgreSQL instance and loads data from an Excel workbook. The script can drop and rebuild the schema using our DDL file, if prompted to do so. It reads each worksheet in the Excel file, which holds clients, media items, books, magazines, digital media, and transactions, parses the rows using Pandas, cleans the values, and inserts them into the matching tables. Inherited tables rely on the existing row in Media_Item and are linked by the shared item_id. For transactions, the script removes any hard-coded primary keys so PostgreSQL can automatically generate them. Each insert is committed individually, and any problem such as duplicate key or failed check constraint is rolled back for that row and written to the console. This approach lets us seed the database with a clean and reliable data set for testing.

Table Contents:

Client:

(1, 'Liam Carter', 'Regular', 'Active', 'liam.carter@yahoo.com', '4155551023')

(2, 'Ava Morgan', 'Regular', 'Active', 'ava.morgan@yahoo.com', '6465558291')

(3, 'Noah Sinclair', 'Student', 'Active', 'noah.sinclair@gmail.com', '3125554407')

(4, 'Maya Patel', 'Student', 'Inactive', 'maya.patel@gmail.com', '7185552678')

(5, 'Ethan Reynolds', 'Senior Citizen', 'Active', 'ethan.reynolds@outlook.com', '2025559473')

(6, 'Zoe Kim', 'Senior Citizen', 'Active', 'zoe.kim@outlook.com', '5035551320')

(7, 'Lucas Bennett', 'Regular', 'Suspended', 'lucas.bennett@ku.edu', '2135553849')

(8, 'Isla Rivera', 'Regular', 'Suspended', 'isla.rivera@ku.edu', '4085556602')

(9, 'Owen Hayes', 'Student', 'Active', 'owen.hayes@hotmail.com', '6175550495')

(10, 'Chloe Wang', 'Other', 'Inactive', 'chloe.wang@hotmail.com', '3475557132')

(11, 'Chris Pratt', 'Regular', 'Active', 'cprat@gmail.com', '5522559321')

(12, 'Daniel Rider', 'Student', 'Active', 'daniel.rider@mail.com', '3165566691')

(13, 'Archie Ryan', 'Regular', 'Inactive', 'notnow@email.com', '2255589612')

(14, 'Jack Black', 'Senior Citizen', 'Suspended', 'jackjackjack@gmail.com', '2014352355')

(15, 'Chloe Smith', 'Regular', 'Active', 'chloe.smith@gmail.com', '3165899998')

(16, 'Charlotte Aitchison', 'Regular', 'Active', 'charlotte2awesome@gmail.com', '9155879874')

(17, 'Kayleigh Amstutz', 'Regular', 'Active', 'kaleigh.amstutz@aol.com', '6978458541')

(18, 'Joseph Harris', 'Senior Citizen', 'Suspended', 'joesph.harris@hotmail.com', '3165699421')

(19, 'Jace Beleren', 'Regular', 'Inactive', 'monoujace@gmail.com', '9645684586')

(20, 'Quintorius Kand', 'Student', 'Suspended', 'q387k563@strixhaven.edu', '2202380250')

(21, 'Wanda Maximoff', 'Regular', 'Active', 'wmax@gmail.com', '8549651235')

(22, 'Harper Ellison', 'Regular', 'Active', 'harper.ellison23@gmail.com', '2025550134')

(23, 'Grayson Tate', 'Senior Citizen', 'Active', 'grayson.tate@yahoo.com', '3035550192')

(24, 'Riley Monroe', 'Regular', 'Inactive', 'riley.m.monroe@outlook.com', '4155550117')

(25, 'Sawyer Smith', 'Student', 'Active', 'sawyer.smith91@gmail.com', '7185550173')

(26, 'Brady Ballinger ', 'Student', 'Active', 'brady.ballinger@yahoo.com', '4075550166')

(27, 'Ian Francis', 'Student', 'Active', 'ian.francis88@outlook.com', '3125550128')

(28, 'Brooke Winslow', 'Other', 'Active', 'brooke.winslow@gmail.com', '6265550145')

(29, 'Saul Goodman', 'Regular', 'Suspended', 'saul.goodman.law@outlook.com', '5058425662')

(30, 'Walter White', 'Senior Citizen', 'Inactive', 'wwhite.chem@gmail.com', '5055550102')

(31, 'Jessie Pinkman', 'Regular', 'Suspended', 'j.pinkman505@yahoo.com', '5055550111')

(32, 'Devin Smith', 'Student', 'Inactive', 'dmith@sunflower.com', '2125557621')

(33, 'Anabelle Wahl', 'Student', 'Active', 'awahl@ku.edu', '7185554917')

(34, 'Alexis Vanderpool', 'Senior Citizen', 'Suspended', 'alexisv@hotmail.com', '3475553845')

(35, 'Mark Ingram', 'Student', 'Inactive', 'mark.ingram29@gmail.com', '4085559347')

(36, 'James Pool', 'Other', 'Suspended', 'james.pool87@yahoo.com', '3125557482')

(37, 'Laura Smith', 'Regular', 'Inactive', 'laura.smith23@outlook.com', '6465551920')

(38, 'Ankus Gray', 'Senior Citizen', 'Active', 'ankus.gray55@gmail.com', '2135558081')

(39, 'Lawrence Outh', 'Regular', 'Suspended', 'outhler@gmail.com', '2025553147')

(40, 'Priyash Gupta', 'Student', 'Active', 'priyash.gupta90@gmail.com', '4155556724')

(41, 'Mike Prince', 'Student', 'Active', 'mike.prince72@hotmail.com', '7185552390')

(42, 'Kingston Mattson', 'Senior Citizen', 'Active', 'kingston.mattson19@aol.com', '5035554419')

(43, 'Kaleb Jones', 'Regular', 'Active', 'kaleb.jones66@yahoo.com', '6175550854')

(44, 'Yui Che', 'Student', 'Inactive', 'yui.che33@protonmail.com', '7025557623')

Transaction:

(1, 1, 106, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 3, 0, 0))

(2, 2, 109, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 5, 0, 0))

(3, 2, 6, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 2, 0, 0))

(4, 7, 110, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 7, 0, 0))

(5, 7, 10, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 10, 0, 0))

(6, 8, 107, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), datetime.datetime(2025, 4, 14, 0, 0))

(7, 3, 54, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), None)

(8, 3, 10, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), None)

(9, 5, 3, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), None)

(10, 6, 56, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), None)

(11, 10, 13, datetime.datetime(2025, 3, 7, 0, 0), datetime.datetime(2025, 3, 15, 0, 0), datetime.datetime(2025, 3, 15, 0, 0))

(12, 10, 19, datetime.datetime(2025, 3, 10, 0, 0), datetime.datetime(2025, 3, 17, 0, 0), datetime.datetime(2025, 3, 19, 0, 0))

(13, 12, 22, datetime.datetime(2025, 3, 20, 0, 0), datetime.datetime(2025, 3, 27, 0, 0), datetime.datetime(2025, 3, 27, 0, 0))

(14, 13, 2, datetime.datetime(2025, 1, 15, 0, 0), datetime.datetime(2025, 1, 22, 0, 0), datetime.datetime(2025, 1, 20, 0, 0))

(15, 15, 14, datetime.datetime(2024, 12, 29, 0, 0), datetime.datetime(2025, 1, 5, 0, 0), datetime.datetime(2025, 1, 15, 0, 0))

(16, 16, 13, datetime.datetime(2025, 3, 16, 0, 0), datetime.datetime(2025, 3, 23, 0, 0), datetime.datetime(2025, 3, 17, 0, 0))

(17, 16, 20, datetime.datetime(2025, 4, 5, 0, 0), datetime.datetime(2025, 4, 12, 0, 0), None)

(18, 16, 11, datetime.datetime(2025, 4, 3, 0, 0), datetime.datetime(2025, 4, 10, 0, 0), None)

(19, 17, 13, datetime.datetime(2025, 3, 18, 0, 0), datetime.datetime(2025, 3, 25, 0, 0), datetime.datetime(2025, 3, 25, 0, 0))

(20, 18, 12, datetime.datetime(2025, 2, 10, 0, 0), datetime.datetime(2025, 2, 17, 0, 0), datetime.datetime(2025, 2, 28, 0, 0))

(21, 21, 13, datetime.datetime(2025, 3, 26, 0, 0), datetime.datetime(2025, 4, 2, 0, 0), None)

(22, 19, 17, datetime.datetime(2025, 4, 2, 0, 0), datetime.datetime(2025, 4, 9, 0, 0), None)

(23, 20, 14, datetime.datetime(2025, 3, 28, 0, 0), datetime.datetime(2025, 4, 4, 0, 0), datetime.datetime(2025, 4, 5, 0, 0))

(24, 20, 9, datetime.datetime(2025, 3, 15, 0, 0), datetime.datetime(2025, 3, 22, 0, 0), datetime.datetime(2025, 3, 21, 0, 0))

(25, 22, 16, datetime.datetime(2025, 4, 1, 0, 0), datetime.datetime(2025, 4, 8, 0, 0), None)

(26, 24, 11, datetime.datetime(2025, 2, 10, 0, 0), datetime.datetime(2025, 2, 17, 0, 0), datetime.datetime(2025, 2, 12, 0, 0))

(27, 25, 12, datetime.datetime(2025, 1, 20, 0, 0), datetime.datetime(2025, 1, 27, 0, 0), datetime.datetime(2025, 1, 25, 0, 0))

(28, 26, 14, datetime.datetime(2025, 3, 10, 0, 0), datetime.datetime(2025, 3, 17, 0, 0), datetime.datetime(2025, 3, 18, 0, 0))

(29, 27, 13, datetime.datetime(2025, 3, 5, 0, 0), datetime.datetime(2025, 3, 12, 0, 0), datetime.datetime(2025, 3, 12, 0, 0))

(30, 28, 10, datetime.datetime(2025, 2, 1, 0, 0), datetime.datetime(2025, 2, 8, 0, 0), datetime.datetime(2025, 2, 7, 0, 0))

(31, 29, 15, datetime.datetime(2025, 3, 25, 0, 0), datetime.datetime(2025, 4, 1, 0, 0), None)

(32, 30, 41, datetime.datetime(2025, 4, 7, 0, 0), datetime.datetime(2025, 4, 14, 0, 0), datetime.datetime(2025, 4, 13, 0, 0))

(33, 31, 33, datetime.datetime(2025, 4, 7, 0, 0), datetime.datetime(2025, 4, 14, 0, 0), datetime.datetime(2025, 4, 12, 0, 0))

(34, 31, 32, datetime.datetime(2025, 3, 7, 0, 0), datetime.datetime(2025, 3, 14, 0, 0), datetime.datetime(2025, 3, 11, 0, 0))

(35, 32, 132, datetime.datetime(2025, 3, 7, 0, 0), datetime.datetime(2025, 3, 14, 0, 0), datetime.datetime(2025, 3, 10, 0, 0))

(36, 30, 44, datetime.datetime(2025, 2, 13, 0, 0), datetime.datetime(2025, 2, 20, 0, 0), datetime.datetime(2025, 2, 19, 0, 0))

(37, 32, 133, datetime.datetime(2025, 2, 13, 0, 0), datetime.datetime(2025, 2, 20, 0, 0), datetime.datetime(2025, 2, 20, 0, 0))

(38, 33, 135, datetime.datetime(2024, 1, 31, 0, 0), datetime.datetime(2024, 2, 7, 0, 0), datetime.datetime(2024, 2, 2, 0, 0))

(39, 34, 134, datetime.datetime(2024, 1, 31, 0, 0), datetime.datetime(2024, 2, 7, 0, 0), datetime.datetime(2024, 2, 2, 0, 0))

(40, 33, 83, datetime.datetime(2023, 3, 20, 0, 0), datetime.datetime(2023, 3, 27, 0, 0), datetime.datetime(2023, 3, 26, 0, 0))

(41, 34, 82, datetime.datetime(2023, 3, 20, 0, 0), datetime.datetime(2023, 3, 27, 0, 0), datetime.datetime(2023, 3, 21, 0, 0))

(42, 37, 89, datetime.datetime(2024, 9, 29, 0, 0), datetime.datetime(2024, 10, 6, 0, 0), datetime.datetime(2024, 10, 5, 0, 0))

(43, 41, 35, datetime.datetime(2024, 2, 17, 0, 0), datetime.datetime(2024, 2, 24, 0, 0), datetime.datetime(2024, 2, 24, 0, 0))

(44, 36, 94, datetime.datetime(2024, 2, 27, 0, 0), datetime.datetime(2024, 3, 5, 0, 0), datetime.datetime(2024, 3, 6, 0, 0))

(45, 44, 38, datetime.datetime(2023, 7, 16, 0, 0), datetime.datetime(2023, 7, 23, 0, 0), datetime.datetime(2023, 7, 24, 0, 0))

(46, 38, 86, datetime.datetime(2025, 3, 9, 0, 0), datetime.datetime(2025, 3, 16, 0, 0), datetime.datetime(2025, 3, 15, 0, 0))

(47, 40, 92, datetime.datetime(2024, 9, 4, 0, 0), datetime.datetime(2024, 9, 11, 0, 0), datetime.datetime(2024, 9, 11, 0, 0))

(48, 35, 31, datetime.datetime(2022, 7, 20, 0, 0), datetime.datetime(2022, 7, 27, 0, 0), datetime.datetime(2022, 7, 28, 0, 0))

(49, 43, 87, datetime.datetime(2022, 7, 18, 0, 0), datetime.datetime(2022, 7, 25, 0, 0), datetime.datetime(2022, 7, 25, 0, 0))

(50, 39, 40, datetime.datetime(2024, 3, 7, 0, 0), datetime.datetime(2024, 3, 14, 0, 0), datetime.datetime(2024, 3, 13, 0, 0))

(51, 42, 84, datetime.datetime(2023, 1, 13, 0, 0), datetime.datetime(2023, 1, 20, 0, 0), datetime.datetime(2023, 1, 20, 0, 0))

Media_item:

(1, 'Available')

(2, 'Available')

(3, 'Unavailable')

(4, 'Available')

(5, 'Available')

(6, 'Available')

(7, 'Available')

(8, 'Available')

(9, 'Available')

(10, 'Unavailable')

(11, 'Unavailable')

(12, 'Available')

(13, 'Unavailable')

(14, 'Available')

(15, 'Unavailable')

(16, 'Unavailable')

(17, 'Unavailable')

(18, 'Available')

(19, 'Available')

(20, 'Unavailable')

(21, 'Available')

(22, 'Available')

(23, 'Available')

(24, 'Available')

(25, 'Available')

(26, 'Available')

(27, 'Available')

(28, 'Available')

(29, 'Available')

(30, 'Available')

(31, 'Available')

(32, 'Available')

(33, 'Available')

(34, 'Available')

(35, 'Available')

(36, 'Available')

(37, 'Available')

(38, 'Available')

(39, 'Available')

(40, 'Available')

(41, 'Available')

(42, 'Available')

(43, 'Available')

(44, 'Available')

(45, 'Available')

(46, 'Available')

(47, 'Available')

(48, 'Available')

(49, 'Available')

(50, 'Available')

(51, 'Available')

(52, 'Available')

(53, 'Available')

(54, 'Unavailable')

(55, 'Available')

(56, 'Unavailable')

(57, 'Available')

(58, 'Available')

(59, 'Available')

(60, 'Available')

(61, 'Available')

(62, 'Available')

(63, 'Available')

(64, 'Available')

(65, 'Available')

(66, 'Available')

(67, 'Available')

(68, 'Available')

(69, 'Available')

(70, 'Available')

(71, 'Available')

(72, 'Available')

(73, 'Available')

(74, 'Available')

(75, 'Available')

(76, 'Available')

(77, 'Available')

(78, 'Available')

(79, 'Available')

(80, 'Available')

(81, 'Available')

(82, 'Available')

(83, 'Available')

(84, 'Available')

(85, 'Available')

(86, 'Available')

(87, 'Available')

(88, 'Available')

(89, 'Available')

(90, 'Available')

(91, 'Available')

(92, 'Available')

(93, 'Available')

(94, 'Available')

(95, 'Available')

(96, 'Available')

(97, 'Available')

(98, 'Available')

(99, 'Available')

(100, 'Available')

(101, 'Available')

(102, 'Available')

(103, 'Available')

(104, 'Available')

(105, 'Available')

(106, 'Available')

(107, 'Available')

(108, 'Available')

(109, 'Available')

(110, 'Available')

(111, 'Available')

(112, 'Available')

(113, 'Available')

(114, 'Available')

(115, 'Available')

(116, 'Available')

(117, 'Available')

(118, 'Available')

(119, 'Available')

(120, 'Available')

(121, 'Available')

(122, 'Available')

(123, 'Available')

(124, 'Available')

(125, 'Available')

(126, 'Available')

(127, 'Available')

(128, 'Available')

(129, 'Available')

(130, 'Available')

(131, 'Available')

(132, 'Available')

(133, 'Available')

(134, 'Available')

(135, 'Available')

(136, 'Available')

(137, 'Available')

(138, 'Available')

(139, 'Available')

(140, 'Available')

(141, 'Available')

(142, 'Available')

(143, 'Available')

(144, 'Available')

(145, 'Available')

(146, 'Available')

(147, 'Available')

(148, 'Available')

(149, 'Available')

(150, 'Available')

Book:

(1, "Harry Potter and the Sorcerer's Stone", 'J. K. Rowling', '1781100489', 'Fiction', 2015)

(2, 'Guitar Notes', 'Mary Amato', '1606841246', 'Romance', 2012)

(3, 'Lawn Boy', 'Gary Paulsen', '553494651', 'Realistic Fiction', 2007)

(4, 'The Phantom Tollbooth', 'Norton Juster', '394815009', 'Fiction', 1961)

(5, 'The BFG', 'Roald Dahl', '224020404', 'Fiction', 1982)

(6, 'The Evolution of Cooperation', 'Robert Axelrod', '465005640', 'Nonfiction', 1984)

(7, 'The Phoenix Project', 'Gene Kim, Kevin Behr, George Spafford', '988262592', 'Fiction', 2013)

(8, 'The Cat in the Hat', 'Dr. Seuss', '717260593', 'Children', 1957)

(9, 'The Awakening', 'Kate Chopin', '486277860', 'Feminist', 1899)

(10, 'Hatchet', 'Gary Paulsen', '27701301', 'Fiction', 1987)

(11, 'Harry Potter and the Chamber of Secrets', 'J. K. Rowling', '439064864', 'Fiction', 1999)

(12, 'The Alchemist', 'Paulo Coelho', '9780061122415', 'Fiction', 1993)

(13, 'Darkhold', 'Chthon', '274154666', 'Nonfiction', None)

(14, 'The Catcher in the Rye', 'J. D. Salinger', '316769487', 'Realistic Fiction', 1991)

(15, 'The Great Gatsby', 'F. Scott Fitzgeral', '743273567', 'Realistic Fiction', 2004)

(16, 'The Crucible', 'Arthur Miller', '140481389', 'Historical Fiction', 1976)

(17, 'The Book Thief', 'Markus Zusak', '375842209', 'Historical Fiction', 2007)

(18, 'Sapiens: A Brief History of Humankind', 'Yuval Noah Harari', '9780062316097', 'Nonfiction', 2015)

(19, 'Wonder', 'R. J. Palacio', '9780375869020', 'Fiction', 2012)

(20, 'The Scarlet Letter', 'Nathaniel Hawthorne', '9780451531353', 'Historical Fiction', 2009)

(21, 'The Silent Code', 'Amanda Leigh', '9780143127741', 'Thriller', 2021)

(22, 'Algorithmic Hearts', 'Cole Everett', '9780062963673', 'Romance', 2020)

(23, 'Cloud Atlas', 'David Mitchell', '9780375507250', 'Science Fiction', 2004)

(24, 'Voices in the Fog', 'Mira Lane', '9780525558882', 'Mystery', 2018)

(25, 'Echoes of Tomorrow', 'J. T. Corbin', '9781982137274', 'Dystopian', 2022)

(26, 'A Brief History of Nearly Everything', 'Bill Bryson', '9780767908184', 'Nonfiction', 2003)

(27, 'Red, White & Royal Blue', 'Casey McQuiston', '9781250316776', 'Romance', 2019)

(28, 'Shadows of the Mind', 'Roger Penrose', '9780195101072', 'Philosophy', 1994)

(29, 'The Art of Fielding', 'Chad Harbach', '9780316126670', 'Realistic Fiction', 2011)

(30, 'Tales from the Loop', 'Simon Stålenhag', '9781982150716', 'Science Fiction', 2020)

(31, 'The Anxious Generation', 'Jonathan Haidt', '9780593655030', 'Self-help', 2024)

(32, 'Outliers', 'Malcolm Gladwell', '9780316017930', 'Nonfiction', 2008)

(33, 'Tuesdays with Morrie', 'Mitch Albom', '9780767905923', 'Biographical', 1997)

(34, 'A Tale of Two Cities', 'Charles Dickens', '9780451530578', 'Historical Fiction', 1859)

(35, 'The Maze Runner', 'James Dashner', '9780385737951', 'Science Fiction', 2009)

(36, 'Beyond Good and Evil', 'Friedrich Nietzsche', '9781503250888', 'Drama', 1886)

(37, 'Brave new world', 'Aldous Huxley', '9780060850524', 'Science Fiction', 1932)

(38, 'Killing Floor', 'Lee Child', '9780515153651', 'Thriller', 1997)

(39, 'Die Trying', 'Lee Child', '9780515142242', 'Thriller', 1998)

(40, 'To Kill a Mocking Bird', 'Harper Lee', '9780060935467', 'Fiction', 1960)

(41, 'The Dark Tower', 'Stephen King', '9781880418628', 'Fantasy', 2004)

(42, "A Handmaid's Tale", 'Margaret Atwood', '9780385539241', 'Science Fiction', 1985)

(43, 'Lord of the Flies', 'William Golding', '9781573226127', 'Psychological Fiction', 1954)

(44, 'Alias Grace', 'Margaret Atwood', '9780385475716', 'Historical Fiction', 1996)

(45, 'The Hunger Games', 'Suzanne Collins', '9780439023528', 'Dystopian', 2008)

(46, 'Beasts of Prey', 'Ayana Gray', '9780593405680', 'Fantasy', 2021)

(47, 'The Girl with the Dragon Tattoo', 'Stieg Larsson', '9780307454546', 'Thriller', 2008)

(48, 'Half Bad', 'Sally Green', '9780670016785', 'Fantasy', 2014)

(49, 'The Dressmaker', 'Rosalie Ham', '9781875989706', 'Gothic Fiction', 2000)

(50, 'The Midnight Library', 'Matt Haig', '9780525559476', 'Fiction', 2020)

Magazine:

(51, 'Tech Horizons', datetime.date(2023, 6, 15), 134)

(52, 'Healthwise Weekly', datetime.date(2024, 2, 10), 101)

(53, 'National Geographic', datetime.date(2023, 7, 12), 39)

(54, 'TIME Magazine', datetime.date(2024, 1, 22), 413)

(55, 'People Magazine', datetime.date(2025, 4, 14), 512)

(56, 'Chicago Tribune', datetime.date(2015, 3, 12), 378)

(57, 'Forbes', datetime.date(2011, 2, 28), 54)

(58, 'Sports Illustrated', datetime.date(2016, 10, 7), 93)

(59, 'Costco Connection', datetime.date(2013, 6, 30), 16)

(60, 'Food Network Magazine', datetime.date(2009, 9, 22), 286)

(61, 'TIME Magazine', datetime.date(2025, 1, 12), 470)

(62, 'The New Yorker', datetime.date(2024, 3, 11), 6743)

(63, 'Wired', datetime.date(2024, 6, 10), 3206)

(64, 'The Atlantic', datetime.date(2024, 2, 1), 1122)

(65, 'Forbes', datetime.date(2024, 8, 20), 350)

(66, 'Rolling Stone', datetime.date(2017, 3, 22), 561)

(67, 'Entertainment Weekly', datetime.date(2010, 11, 15), 1250)

(68, 'The Economist', datetime.date(2020, 12, 2), 7510)

(69, 'New Scientist', datetime.date(2024, 5, 18), 3437)

(70, "Reader's Digest", datetime.date(2024, 6, 1), 2984)

(71, 'Scientific American', datetime.date(2025, 4, 1), 8812)

(72, 'Better Homes & Gardens', datetime.date(2024, 3, 20), 1556)

(73, 'National Geographic Kids', datetime.date(2023, 7, 25), 310)

(74, 'Popular Mechanics', datetime.date(2022, 5, 10), 4412)

(75, "Men's Health", datetime.date(2023, 1, 5), 927)

(76, 'Vanity Fair', datetime.date(2024, 9, 18), 2133)

(77, 'GQ Magazine', datetime.date(2023, 8, 8), 1625)

(78, 'Smithsonian', datetime.date(2022, 10, 14), 1187)

(79, 'Consumer Reports', datetime.date(2023, 12, 5), 5720)

(80, 'Real Simple', datetime.date(2024, 6, 9), 2417)

(81, 'PC Mag', datetime.date(1982, 1, 31), 1)

(82, 'Science News', datetime.date(2025, 4, 1), 207)

(83, 'MIT Technology Review', datetime.date(2024, 12, 31), 127)

(84, 'Science News', datetime.date(2024, 11, 16), 206)

(85, 'National Geographic', datetime.date(2025, 1, 1), 1)

(86, 'National Geographic', datetime.date(2025, 2, 1), 2)

(87, 'National Geographic', datetime.date(2025, 3, 1), 3)

(88, 'National Geographic', datetime.date(2025, 4, 1), 4)

(89, 'TIME Magazine', datetime.date(2024, 4, 14), 203)

(90, 'The New Yorker', datetime.date(2024, 4, 15), 100)

(91, 'Scientific American ', datetime.date(2024, 4, 1), 330)

(92, 'The Atlantic', datetime.date(2024, 4, 1), 333)

(93, 'Wired', datetime.date(2023, 5, 10), 207)

(94, 'Vogue ', datetime.date(2023, 3, 1), 134)

(95, 'The New York Times Magazine', datetime.date(2024, 3, 3), 102)

(96, 'The Wall Street Journal', datetime.date(2025, 2, 14), 150)

(97, 'Harvard Business Review', datetime.date(2025, 1, 1), 98)

(98, 'The Guardian Weekly', datetime.date(2024, 12, 15), 75)

(99, 'Fast Company', datetime.date(2024, 11, 10), 210)

(100, 'Scientific American Mind', datetime.date(2024, 10, 5), 67)

Digital_media:

(101, 'The Midnight Library', 'Matt Haig', '9780525559498', 'Fiction', 2020)

(102, 'Where the Crawdads Sing', 'Delia Owens', '9780735219111', 'Fiction', 2018)

(103, 'Becoming', 'Michelle Obama', '9781524763152', 'Biography', 2018)

(104, 'It Ends With Us', 'Colleen Hoover', '9781501110368', 'Romance', 2016)

(105, 'The Silent Patient', 'Alex Michaelides', '9781250301710', 'Thriller', 2019)

(106, 'The Kite Runner', 'Khaled Hosseini', '9781594631931', 'Fiction', 2003)

(107, 'The Da Vinci Code', 'Dan Brown', '9780307474278', 'Thriller', 2003)

(108, 'Harry Potter and the Goblet of Fire', 'J. K. Rowling', '9780439139601', 'Fantasy', 2000)

(109, '1984', 'George Orwell', '9780547249643', 'Dystopian', 2013)

(110, 'Pride and Prejudice', 'Jane Austen', '9781101099063', 'Fantasy', 2008)

(111, 'The Hobbit', 'J.R.R. Tolkien', '9780007322602', 'Fantasy', 2010)

(112, 'The Night Circus', 'Erin Morgenstern', '9780307744432', 'Fantasy', 2011)

(113, 'Educated', 'Tara Westover', '9780399590504', 'Biography', 2018)

(114, 'Circe', 'Madeline Miller', '9780316556347', 'Mythology', 2018)

(115, 'Project Hail Mary', 'Andy Weir', '9780593135204', 'Science Fiction', 2021)

(116, 'Verity', 'Colleen Hoover', '9781791392796', 'Thriller', 2018)

(117, 'The Four Agreements', 'Don Miguel Ruiz', '9781878424310', 'Self-Help', 1997)

(118, 'A Man Called Ove', 'Fredrik Backman', '9781476738024', 'Fiction', 2014)

(119, 'Reminders of Him', 'Colleen Hoover', '9781542025607', 'Romance', 2022)

(120, 'Atomic Habits', 'James Clear', '9780735211290', 'Self-Help', 2018)

(121, 'Normal People', 'Sally Rooney', '9781984822185', 'Fiction', 2018)

(122, 'Before We Were Strangers', 'Renee Carlino', '9781501105777', 'Romance', 2015)

(123, 'Thinking, Fast and Slow', 'Daniel Kahneman', '9780374533557', 'Nonfiction', 2011)

(124, 'The Paper Palace', 'Miranda Heller', '9780593329825', 'Fiction', 2021)

(125, 'Who Will Carry The Boats?', 'David Goggins', '9781544512273', 'Biography', 2018)

(126, 'The Paris Library', 'Janet Skeslien Charles', '9781982134198', 'Historical Fiction', 2021)

(127, 'The Seven Husbands of Evelyn Hugo', 'Taylor Jenkins Reid', '9781501161933', 'Fiction', 2017)

(128, 'The Land Before Time', 'Don Bluth', '9780783233584', 'Animation', 1988)

(129, 'The Land Before Time II: The Great Valley Adventure', 'Roy Allen Smith', '9780783233591', 'Animation', 1994)

(130, 'The Land Before Time III: The Time of the Great Giving', 'Roy Allen Smith', '9780783233607', 'Animation', 1995)

(131, 'The Land Before Time IV: Journey Through the Mists', 'Roy Allen Smith', '9780783233614', 'Animation', 1996)

(132, 'The Shining', 'Stephen King', '9783785746042', 'Horror', None)

(133, 'The Dead Zone', 'Stephen King', '9788401342769', 'Thriller', None)

(134, 'Misery', 'Stephen King', '9789520405540', 'Horror', None)

(135, "Gerald's Game", 'Stephen King', '9788401354151', 'Horror', None)

(136, 'Fourth Wing', 'Rebecca Yarros', '9781649374042', 'Fantasy', 2023)

(137, 'Happy Place', 'Emily Henry', '9780593441275', 'Romance', 2023)

(138, 'Yellowface', 'R.F Kuang', '9780063250833', 'Fiction', 2023)

(139, 'Iron Flame', 'Rebecca Yarros', '9781649374172', 'Fantasy', 2023)

(140, 'Tomorrow, and Tomorrow, and Tomorrow', 'Gabrielle Zevin', '9780593321201', 'Fiction', 2022)

(141, 'The Woman in Me', 'Britney Spears', '9781668009048', 'Memoir', 2023)

(142, 'Oath and Honor', 'Liz Cheney', '9780316475310', 'Nonfiction', 2023)

(143, 'The Bee Sting', 'Paul Murray', '9780241996975', 'Fiction', 2023)

(144, 'The Heaven & Earth Grocery Store', 'James McBride', '9780593422946', 'Fiction', 2023)

(145, 'Wager', 'David Grann', '9780385534260', 'Nonfiction', 2023)

(146, 'The Women', 'Kristin Hannah', '9781250283788', 'Historical Fiction', 2024)

(147, 'Funny Story', 'Emily Henry', '9780593826287', 'Romance', 2024)

(148, 'James', 'Percival Everett', '9780385549820', 'Fiction', 2024)

(149, 'The Lost Bookshop', 'Evie Woods', '9780008609214', 'Magic Realism', 2023)

(150, 'The Saint of Bright Doors', 'Vajra Chandrasekera', '9781250847386', 'Fantasy', 2023)