WolfCity Publishing House

CSC 540 Database Management Systems
Project 2

Harini Bharata - hbharat@ncsu.edu,
Krishna Saurabh Vankadaru - kvankad@ncsu.edu,
Sri Pallavi Damuluri - sdamulu@ncsu.edu,
Subodh Thota - sthota@ncsu.edu

March 11th, 2022

1. Assumptions

1. There are no special journalists, all journalists are authors.
2. Each article in a publication is written by one author.
3. A distributor can place an order for only one publication but multiple quantities at a time.
4. A book can have more than one author.
5. A distributor can only hold one account, and the current due amount will be tracked in the account.
6. A chapter is a part of exactly one book and similarly, an article is part of exactly one periodical.
7. Chapters in a book cannot be written by different authors, only the book has been authored.
8. Payment to authors and editors will need the input of the payment amount, and we will not store any salary information for the staff.
9. Payment to staff will be made on the salary date, that is when the payment is initiated.
10. collection_date in Payment is allowed to be null.
11. Every member of the staff will have a unique staff id.
12. Every new edition of a book will have a new publication id, similarly, every new periodical will have a new publication id.
13. Each article in a periodical will have a unique title.
14. Topic can be null for a publication.

2. Global Relational Database Schema:

Editors(<u>staff_ID</u>, type)

<u>staff_ID</u> → <u>staff_ID</u>, type

The above holds because each Editor has a unique staff_ID Hence, each Editor can be identified using a staff_ID.

Since the left-hand side of the above relation belongs to a superkey, it is said to be in BCNF.

Since the relation is in BCNF, it is also in 3NF. No other attribute or combination of attributes allows us to uniquely identify an editor. Therefore, no other functional dependency holds.

Authors(<u>staff_ID</u>, type)

<u>staff_ID</u> → <u>staff_ID</u>, type

The above holds because each Author has a unique staff_ID Hence, each Editor can be identified using a staff_ID.

Since the left-hand side of the above relation belongs to a superkey, it is said to be in BCNF. Since the relation is in BCNF, it is also in 3NF. No other attribute or combination of attributes allows us to uniquely identify an author. Therefore, no other functional dependency holds.

Edits(staff_id, publication_id)
staff_id, publication_id → staff_id, publication_id
Since both the attributes are in superkey, the relation is in BCNF. Hence, it is in 3NF.

Distributor(account_number, phone, city, street_address, type, name, balance, contact_person)
account_number → account_number, phone, city, street_address, type, name, balance, contact_person
The above holds because each distributor has a unique account_number. Hence, each distributor can be identified using an account_number.
Since the left-hand side of the above relation belongs to a superkey, it is said to be in BCNF. Since the relation is in BCNF, it is also in 3NF. No other attribute or combination of attributes allows us to uniquely identify a Periodical. Therefore, no other functional dependency holds.

Distributorpayments(account_number, payment_date, amount_paid)
account_number, payment_date → account_number, payment_date, amount_paid
A distributor payment can be uniquely identified using account_number and payment_date.
The above relation is in BCNF because the left-hand side is a part of the superkey. Hence, it is said to be in 3NF too.
account_number → account_number, payment_date, amount_paid
The above functional dependency doesn't hold because we cannot identify a distributor's payment uniquely using the account_number only.

Therefore, the only functional dependency that holds is
account_number, payment_date → account_number, payment_date, amount_paid

Orders(order_number,publication_id, distributor_account_number, order_date, order_delivery_date, total_cost, shipping_cost)
order_number→order_number,publication_id, distributor_account_number, order_date, order_delivery_date, total_cost, shipping_cost holds because an order can be uniquely identified using the order_number.
Since the left-hand side of the above relation belongs to a superkey, it is said to be in BCNF. Since the relation is in BCNF, it is also in 3NF. No other attribute or combination of attributes allows us to uniquely identify an Order. Therefore, no other functional dependency holds.

Publications(publication_ID, title, topic, type, price)

publication_ID → publication_ID, title, topic, type, price

The above functional dependency holds because a publication_ID can be uniquely identified using the publication_ID.

The above relation is in BCNF because the left-hand side belongs to the superkey. Hence, it is also in 3NF.

title→ publication_ID, title, topic, type, price doesn't hold because multiple publications can have the same title.

No other attribute or combination of attributes allows us to uniquely identify a Periodical. Therefore, no other functional dependency holds.

Books(publication_ID, ISBN, Edition, publication_date)

publication_ID→publication_ID, ISBN, Edition, publication_date holds because a Book can be uniquely identified using a publication_ID.

This relation is in BCNF because the left-hand side belongs to the superkey. Since it is in BCNF, the relation is also in 3NF.

ISBN →publication_ID, ISBN, Edition, publication_date doesn't hold because we cannot use ISBN to uniquely identify a Book.

Similarly, no other attribute or combination of attributes allows us to uniquely identify a book. Therefore, no other functional dependency holds.

Periodicals(publication_id, issue_date, periodicity)

publication_id → publication_id, issue_date, periodicity holds because a periodical can be uniquely identified using a publication_ID.

This relation is in BCNF since the left-hand side belongs to the superkey. Since it is in BCNF, the relation is also in 3NF. No other attribute or combination of attributes allows us to uniquely identify a Periodical. Therefore, no other functional dependency holds.

Staff(staff_ID, name, role, phone_number)

staff_ID → staff_ID, name, role, phone_number holds because a staff member can be uniquely identified using the staff_ID.

The relation is in BCNF because the left-hand side is in superkey. Hence, it is also in 3NF.

name → role, phone_number doesn't hold because multiple staff members may have the same name. Similarly, no other attribute or combination of attributes allows us to uniquely identify a Staff. Therefore, no other functional dependency holds.

Chapters(publication_id, title, text)
publication_id, title → publication_id, title, text
The above functional dependency holds because a chapter can be uniquely identified using the publication_ID nad title. The above relation is in BCNF because the left-hand side is in superkey. publication_id→ publication_id, title, text don’t hold because a single publication may have multiple chapters in it. Therefore, we will not be able to identify a chapter uniquely. Similarly, no other attribute or combination of attributes allows us to uniquely identify a chapter. Therefore, no other functional dependency holds.

Articles(publication_id, title, text, creation_date, topic)
publication_id, title → publication_id, title, text, creation_date, topic
The above functional dependency holds because an article can be uniquely identified using the publication_ID nad title. The above relation is in BCNF because the left-hand side is in superkey. No other attribute or combination of attributes allows us to uniquely identify an article. Therefore, no other functional dependency holds.

WritesBook(staff_id, publication_id)
staff_id, publication_id → staff_id, publication_id
Since both the attributes are in superkey, the relation is in BCNF. Hence, it is in 3NF.

WritesArticles(staff_id, publication_id, title)
staff_id, publication_id, title → staff_id, publication_id, title
Since all the attributes are in superkey, the relation is in BCNF. Hence, it is in 3NF.

Payment(staff_ID, salary_date, payment_amount, collection_date)
staff_ID, salary_date → staff_ID, salary_date, payment_amount, collection_date
The above functional dependency holds because a payment record can be uniquely identified using staff_ID and salary_date. The above relation is in BCNF because the left-hand side is in superkey. No other attribute or combination of attributes allows us to uniquely identify a payment. Therefore, no other functional dependency holds.

3. Design for Global Schema:

Design decision for global schema

The design of the relational schemas is derived from the Entity-Relationship diagrams mentioned in the previous project report 1. We have used the E/R method to convert the hierarchical relationships of Staff, Authors, and Editors. And also for different kinds of publications, books, and periodicals. This method helps reduce NULL values and redundancy in the tables.

Staff(staff_ID, name, role, phone_number)

- staff_ID is the primary key for the staff relation schema, and it is enforced to be not null.
- name, role, and phone_number are required attributes for staff and are enforced to be non-null.

Editors(staff_ID, type)

- staff_ID is the primary key for the Editors relation schema, and it is enforced to be not null.
- staff_ID is the foreign key that references staff_ID from the Staff relation.
- The foreign key follows cascading referential integrity on both delete and update operations.
- the type must have a value, it cannot be null as it identifies whether the author is invited or staff.

Authors(staff_ID, type)

- staff_ID is the primary key for the Authors relation schema, and it is enforced to be not null.
- staff_ID is the foreign key that references staff_ID from the Staff relation.
- The foreign key follows cascading referential integrity on both delete and update operations.
- the type must have a value, it cannot be null as it identifies whether the author is invited or staff.

Edits(staff_ID, publication_ID)

- staff_ID and publication_ID together are the composite primary keys for the relation Edits. They are enforced to be non NULL.
- staff_ID is the foreign key that references staff_ID from the Editors relation.
- The staff_ID foreign key follows cascading referential integrity on both delete and update operations.

- publication_ID is the foreign key that references publication_ID from the Publications relation.
- The publication_ID foreign key follows cascading referential integrity on both delete and updates operations.

Publications(publication_ID, title, topic, type, price)

- publication_ID is the primary key for the Publications relation schema, and it is enforced to be not null.
- title, type, and price cannot be null as there are required to define a Publication.
- **the topic can be null as it is possible that the topic may not be known during the time of creation.**

Books(publication_ID, ISBN, Edition, publication_date)

- publication_ID is the primary key for the Publications relation schema, and it is enforced to be not null.
- ISBN is unique for all books and can uniquely identify entries in the table, hence it is a candidate key. ISBN is enforced to be not NULL, and UNIQUE.
- The edition is allowed to be NULL as a book can be present without an edition number. A null value for the Edition attribute means that we don't know the edition number for the book. It is not required to have an edition number to store the book details.
- Publication_date cannot be null as it is required to store the details of a book publication.
- publication_ID is the foreign key that references publication_ID from the Publications relation.
- The foreign key publication_ID follows cascading referential integrity on both delete and update operations.

Periodicals(publication_ID, issue_date, periodicity)

- publication_ID is the primary key for the Periodicals relation schema, and it is enforced to be not null.
- publication_ID is the foreign key that references publication_ID from the Publications relation.
- The foreign key follows cascading referential integrity on both delete and update operations.
- issue_date and periodicity cannot be null.

Chapters(publication_ID, title, text)

- publication_ID is the primary key for the Chapters relation schema and is enforced to be not null.
- publication_ID is the foreign key that references publication_ID from the Books relation.

- The foreign key follows cascading referential integrity on both delete and update operations.
- Title and text attributes must be not NULL as they have to exist to create a chapter that is part of a book.

Articles(publication_ID, title, text, creation_date)

- publication_ID and title are the primary keys for the Articles relation schema and are enforced to be not null.
- publication_ID is the foreign key that references publication_ID from the Periodicals relation.
- The foreign key follows cascading referential integrity on both delete and update operations.
- The combination of publication_ID and title must be unique.

WritesBook(staff_ID, publication_ID)

- publication_ID and staff_ID are the primary keys for the WritesBook relation schema and are enforced to be not null.
- staff_ID is the foreign key that references staff_ID from the Authors relation.
- The foreign key staff_ID follows cascading referential integrity on both delete and update operations.
- publication_ID is the foreign key that references publication_ID from the Books relation.
- The foreign key publication_ID follows cascading referential integrity on both delete and update operations.

WritesArticles(staff_ID, publication_ID, title)

- publication_ID, staff_ID, and title are the primary keys for the WritesArticles relation schema and are enforced to be not null.
- staff_ID is the foreign key that references staff_ID from the Authors relation.
- The foreign key staff_ID follows cascading referential integrity on both delete and update operations.
- publication_ID and title are the foreign keys that are referenced from publication_ID and title from the Articles relation.
- The foreign key publication_ID follows cascading referential integrity on both delete and update operations.

Payment(staff_ID, salary_date, payment_amount, collection_date)

- staff_ID and salary_date are the primary keys for the Payment relation schema and are enforced to be not null.
- staff_ID and salary_date together should be unique.
- staff_ID is the foreign key that references staff_ID from the Staff relation.
- The foreign key staff_ID follows cascading referential integrity on both delete and update operations.
- payment_amount cannot be null.
- collection_date is allowed to be null as a staff member may not have collected the payment yet, even though it has been posted. When the payment is collected, collection_date will be updated.

Distributor(account_number, phone, city, street_address, type, name, balance, contact_person)

- acount_number is the primary key for the Distributor relation schema and is enforced to be not null.
- phone, city, street_address, type, name, balance, contact_person cannot be null as these values are required to keep track of the distributor.

DistributorPayments(account_number, payment_date, amount_paid)

- acount_number and payment_date are the primary keys for the DistributorPayments relation schema and are enforced to be not null.
- account_number is the foreign key that references account_number from the Distributor relation.
- account_number and payment_date together must be unique.
- amount_paid cannot be null and should contain a value.

Orders(order_number, publication_ID, distrubutor_account_number, order_date, order_delivery_date, total_cost, shipping_cost)

- Order_number is the primary key for the Orders relation schema and is enforced to be not null.
- Publication_id, distrubutor_account_number, order_date, order_delivery_date, total_cost, shipping_cost are details that are necessary to place an order. Hence they are enforced to be not NULL.

- publication_ID is the foreign key that references publication_ID from the Publications relation.
- The foreign key publication_ID follows cascading referential integrity on both delete and update operations.
- Distrubutor_account_number is the foreign key that references account_number from the Distributors relation.
- The foreign key Distrubutor_account_number follows cascading referential integrity on both delete and update operations.

4. Base Relations

Create table for Publications relation

```
CREATE TABLE IF NOT EXISTS `Publications` (
    `publication_ID` INT,
    `title` VARCHAR(50) NOT NULL,
    `topic` VARCHAR(30),
    `type` VARCHAR(30) NOT NULL,
    `price` DECIMAL(8,2) NOT NULL,
    PRIMARY KEY(`publication_ID`)
);
```

Create table for Staff relation

```
CREATE TABLE IF NOT EXISTS `Staff` (
   `staff_ID` INT,
   `name` VARCHAR(120) NOT NULL,
   `role` VARCHAR(15) NOT NULL,
   `phone_number` VARCHAR(20) NOT NULL,
    PRIMARY KEY(`staff_ID`)
);
```

Create table for Books relation

```
CREATE TABLE IF NOT EXISTS `Books` (
    `publication_ID` INT,
    `ISBN` INT NOT NULL,
    `edition` INT,
    `publication_date` DATE NOT NULL,
     PRIMARY KEY(`publication_ID`),
     UNIQUE(`ISBN`),
     FOREIGN KEY(`publication_ID`) REFERENCES
    `Publications`(`publication_ID`)
     ON DELETE CASCADE
     ON UPDATE CASCADE
);
```

Create table for Edits relation

```
CREATE TABLE IF NOT EXISTS `Edits` (
      `staff_ID` INT,
      `publication_ID` INT,
       PRIMARY KEY (`publication_ID`,`staff_ID`),
       FOREIGN KEY (`publication_ID`) REFERENCES
      `Publications`(`publication_ID`)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);
```

Create table for Payment relation

```
CREATE TABLE IF NOT EXISTS `Payment` (
     `staff_ID` INT,
     `salary_date` DATE NOT NULL,
     `payment_amount` DECIMAL(8,2) NOT NULL,
     `collection_date` date,
      PRIMARY KEY (`staff_ID`,`salary_date`),
      FOREIGN KEY(`staff_ID`) REFERENCES `Staff`(`staff_ID`)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);
```

Create table for Distributors relation

```
CREATE TABLE IF NOT EXISTS `Distributors` (
     `account_number` INT,
     `phone` VARCHAR(255) NOT NULL,
     `city` VARCHAR(255) NOT NULL,
     `street_address` VARCHAR(255) NOT NULL,
     `type` VARCHAR(255) NOT NULL,
     `name` VARCHAR(255) NOT NULL,
     `balance` INT NOT NULL,
     `contact_person` VARCHAR(255) NOT NULL,
      PRIMARY KEY(`account_number`)
);
```

Create table for DistributorPayments relation

```
CREATE TABLE IF NOT EXISTS `DistributorPayments` (
     `account_number` INT,
     `payment_date` DATE NOT NULL,
     `amount_paid` DECIMAL(8,2) NOT NULL,
      PRIMARY KEY (`account_number`,`payment_date`)
     FOREIGN          KEY          (account_number)          REFERENCES
Distributors(account_number)
      ON DELETE CASCADE
```

```
       ON UPDATE CASCADE

);
```

Create table for Orders relation

```
CREATE TABLE IF NOT EXISTS `Orders` (
     `order_number` INT ,
     `publication_ID` INT ,
     `distributor_account_no` INT,
     `order_date` DATE NOT NULL,
     `order_delivery_date` DATE NOT NULL,
     `number_of_copies` INT NOT NULL,
     `total_cost` DECIMAL(8,2) NOT NULL,
     `shipping_cost` DECIMAL(8,2) NOT NULL,
     PRIMARY KEY (`order_number`),
     FOREIGN KEY (publication_ID) REFERENCES
     Publications(publication_ID)
      ON DELETE CASCADE
     ON UPDATE CASCADE,
     FOREIGN KEY (distributor_account_no) REFERENCES
     Distributors(account_number)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

Create table for Editors relation

```
CREATE TABLE IF NOT EXISTS  `Editors` (
     `staff_ID` INT,
     `type` varchar(30) NOT NULL,
      PRIMARY KEY(`staff_ID`),
      FOREIGN KEY (`staff_ID`) REFERENCES `Staff` (`staff_ID`)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);
```

Create table for Authors relation

```
CREATE TABLE IF NOT EXISTS `Authors` (
   `staff_ID` INT,
   `type` varchar(30) NOT NULL,
    PRIMARY KEY(`staff_ID`),
    FOREIGN KEY (`staff_ID`) REFERENCES `Staff` (`staff_ID`)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

Create table for Periodicals relation

```
CREATE TABLE IF NOT EXISTS `Periodicals` (
    `publication_id` INT,
    `issue_date` DATE NOT NULL,
    `periodicity` VARCHAR(30) NOT NULL,
    PRIMARY KEY (`publication_ID`),
     FOREIGN KEY (`publication_ID`) REFERENCES
`Publications`(`publication_ID`)
     ON DELETE CASCADE
     ON UPDATE CASCADE
);
```

Create table for Chapters relation

```
CREATE TABLE IF NOT EXISTS `Chapters` (
   `publication_ID` INT,
   `title` VARCHAR(30) NOT NULL,
   `text` TEXT NOT NULL,
   PRIMARY KEY (`publication_ID`,`title`),
   FOREIGN KEY (`publication_ID`) REFERENCES `Books`(`publication_ID`)
   ON DELETE CASCADE ON UPDATE CASCADE
);
```

Create table for WritesBook relation

```
CREATE TABLE IF NOT EXISTS `WritesBook` (
     `staff_ID` INT,
      `publication_ID` INT,
       PRIMARY KEY(`staff_ID`,`publication_ID`),
    FOREIGN KEY(`publication_ID`) REFERENCES `Books`(`publication_ID`)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
       FOREIGN KEY(`staff_ID`) REFERENCES `Authors`(`staff_ID`)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);
```

Create table for Articles relation

```
CREATE TABLE IF NOT EXISTS `Articles` (
     `publication_ID` INT,
     `title` VARCHAR(50) NOT NULL,
     `text` TEXT NOT NULL,
     `creation_date` DATE NOT NULL,
      PRIMARY KEY(`publication_ID`,`title`),
                      FOREIGN     KEY(`publication_ID`)     REFERENCES
`Periodicals`(`publication_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

Create table for WritesArticles relation

```
CREATE TABLE IF NOT EXISTS `WritesArticles` (
      `staff_ID` INT,
      `publication_ID` INT,
      `title` VARCHAR(50),
       PRIMARY KEY(`staff_ID`,`publication_ID`, `title`),
      FOREIGN      KEY(`publication_ID`,      `title`)      REFERENCES
`Articles`(`publication_ID`, `title`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
      FOREIGN KEY(`staff_ID`) REFERENCES `Authors`(`staff_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```
SELECT * FROM Publications;

+----------------+----------------------------+------------+------------+-------+
| publication_ID | title                      | topic      | type       | price |
+----------------+----------------------------+------------+------------+-------+
|           1001 | Brain Science              | Science    | Book       | 23.00 |
|           1002 | Animal Fashion             | Fashion    | Book       | 40.00 |
|           1003 | Introduction to Blockchain | Technology | Book       | 48.00 |
|           1004 | Introduction to Food       | Health     | Book       | 24.00 |
|           1005 | Science Today              | Science    | Periodical | 34.00 |
|           1006 | Health Today               | Health     | Periodical | 24.00 |
|           1007 | Fashion Today              | Fashion    | Periodical | 44.00 |
|           1008 | Technology Everyday        | Technology | Periodical | 18.00 |
|           1009 | New Science                | Science    | Book       | 25.00 |
|           1012 | Fitness                    | Science    | Book       | 65.00 |
+----------------+----------------------------+------------+------------+-------+
```

```
SELECT * FROM Staff;

+----------+---------+--------+--------------+
| staff_ID | name    | role   | phone_number |
+----------+---------+--------+--------------+
|     6991 | Subodh  | Admin  | 9391234560   |
|     6992 | Pallavi | Author | 9391234561   |
|     6993 | Charles | Editor | 9391234562   |
|     6994 | Saurab  | Author | 9391234563   |
|     6995 | Harish  | Editor | 9391234564   |
|     6996 | Eshwar  | Author | 9391234565   |
|     6997 | Sandeep | Editor | 9391234566   |
|     6998 | Harika  | Author | 9391234567   |
|     6999 | Bhavya  | Editor | 9391234568   |
+----------+---------+--------+--------------+
```

```
SELECT * FROM Books;
+----------------+--------+---------+------------------+
| publication_ID | ISBN   | edition | publication_date |
+----------------+--------+---------+------------------+
|           1001 | 123456 |       1 | 2022-02-02       |
|           1002 | 123457 |       1 | 2022-04-03       |
|           1003 | 123458 |       1 | 2022-03-02       |
|           1004 | 123459 |       1 | 2022-04-12       |
+----------------+--------+---------+------------------+
```

SELECT * FROM Orders;

```
+--------------+----------------+------------------------+------------+---------------------+-------
-----------+------------+---------------+
| order_number | publication_ID | distributor_account_no | order_date | order_delivery_date |
number_of_copies | total_cost | shipping_cost |
+--------------+----------------+------------------------+------------+---------------------+-------
-----------+------------+---------------+
|        13345 |           1002 |                   3001 | 2021-03-02 | 2021-03-08          |
10 |     400.00 |         20.00 |
|        13346 |           1003 |                   3002 | 2021-03-02 | 2021-03-08          |
10 |     480.00 |         20.00 |
|        13347 |           1004 |                   3003 | 2021-04-02 | 2021-04-08          |
10 |     240.00 |         20.00 |
|        13349 |           1003 |                   3002 | 2021-05-02 | 2021-05-08          |
10 |     480.00 |         20.00 |
|        13350 |           1004 |                   3003 | 2021-06-02 | 2021-06-08          |
10 |     240.00 |         20.00 |
|        13352 |           1003 |                   3002 | 2021-05-02 | 2021-05-08          |
10 |     480.00 |         20.00 |
|        13353 |           1002 |                   3001 | 2021-03-06 | 2021-04-08          |
5 |     200.00 |         20.00 |
|        13354 |           1003 |                   3001 | 2021-03-15 | 2021-05-08          |
5 |     240.00 |         20.00 |
+--------------+----------------+------------------------+------------+---------------------+-------
-----------+------------+---------------+
```

SELECT * FROM Edits;

```
+----------+----------------+
| staff_ID | publication_ID |
+----------+----------------+
|     6993 |           1001 |
|     6995 |           1001 |
|     6999 |           1001 |
|     6995 |           1002 |
|     6997 |           1003 |
|     6993 |           1004 |
|     6997 |           1004 |
|     6993 |           1006 |
|     6997 |           1006 |
|     6995 |           1007 |
|     6999 |           1007 |
|     6993 |           1008 |
|     6995 |           1009 |
+----------+----------------+
SELECT * FROM Payments;
+----------+-------------+----------------+-----------------+
| staff_ID | salary_date | payment_amount | collection_date |
+----------+-------------+----------------+-----------------+
|     6992 | 2021-03-02  |         900.02 | 2021-03-10      |
|     6992 | 2021-04-02  |         900.02 | 2021-04-10      |
|     6994 | 2021-03-02  |         600.02 | 2021-03-08      |
|     6995 | 2021-04-02  |          89.02 | 2021-04-10      |
|     6996 | 2021-03-02  |         900.02 | 2021-03-10      |
|     6996 | 2021-04-02  |         900.02 | 2021-04-10      |
|     6997 | 2021-04-02  |          89.02 | 2021-04-10      |
|     6998 | 2021-03-02  |         600.02 | 2021-03-08      |
+----------+-------------+----------------+-----------------+
```

```
SELECT * FROM Distributors;
+----------------+------------+----------+------------------------+-----------+--------------+---------+----------------+
| account_number | phone      | city     | street_address         | type      | name         | balance | contact_person |
+----------------+------------+----------+------------------------+-----------+--------------+---------+----------------+
|           3001 | 9951009755 | Cary     | 111, Avent Ferry Road  | library   | Distributor9 |       0 | Shane          |
|           3002 | 9102222222 | New York | 222, Brooke Street     | library   | Distributor2 |       0 | Natalie        |
|           3003 | 9103333333 | Chicago  | 333, High Point Ave    | wholesale | Distributor3 |       0 | Jarette        |
|           3004 | 9104444444 | Boston   | 444, Silver Spear Lane | wholesale | Distributor4 |       0 | Shaina         |
|           3005 | 9105555555 | Austin   | 555, Second Street     | bookstore | Distributor5 |       0 | Mallory        |
|           3006 | 9106666666 | Raleigh  | 666, Walnutwood        | bookstore | Distributor6 |       0 | Eric           |
|           3007 | 9107777777 | Chicago  | 777, North 6th Lane    | library   | Distributor7 |       0 | Deepti         |
|           3008 | 9108888888 | Austin   | 888, Third Street      | bookstore | Distributor8 |       0 | Shake          |
+----------------+------------+----------+------------------------+-----------+--------------+---------+----------------+
```

```
SELECT * FROM DistributorPayments;

+----------------+--------------+-------------+
| account_number | payment_date | amount_paid |
+----------------+--------------+-------------+
|           3001 | 2021-01-01   |      790.00 |
|           3001 | 2021-01-02   |      420.00 |
|           3002 | 2021-02-03   |      122.00 |
|           3003 | 2021-07-09   |     1899.00 |
|           3004 | 2021-01-02   |      790.00 |
|           3005 | 2021-09-01   |      987.00 |
|           3006 | 2021-08-01   |     1001.00 |
|           3007 | 2021-02-02   |      798.60 |
|           3008 | 2021-06-08   |      999.00 |
+----------------+--------------+-------------+
SELECT * FROM Editors;
+----------+---------+
| staff_ID | type    |
+----------+---------+
|     6993 | Staff   |
|     6995 | Invited |
|     6997 | Staff   |
|     6999 | Invited |
+----------+---------+
```

```
SELECT * FROM Authors;
+----------+---------+
| staff_ID | type    |
+----------+---------+
|     6992 | Staff   |
|     6994 | Invited |
|     6996 | Staff   |
|     6998 | Invited |
+----------+---------+
```

```
SELECT * FROM Periodicals;
+----------------+------------+-------------+
| publication_id | issue_date | periodicity |
+----------------+------------+-------------+
|           1005 | 2022-01-12 | Monthly     |
|           1006 | 2022-04-02 | Yearly      |
|           1007 | 2022-03-10 | Monthly     |
|           1008 | 2022-09-03 | Weekly      |
+----------------+------------+-------------+
```

```
SELECT * FROM Chapters;
+----------------+----------------------------+----------------------------------+
| publication_ID | title                      | text                             |
+----------------+----------------------------+----------------------------------+
|           1001 | Brain Science              | Brain Science Text1              |
|           1001 | Science Direct             | Science Today                    |
|           1002 | Animal Fashion             | Animal Fashion Text1             |
|           1003 | Introduction to Blockchain | Introduction to Blockchain Text1 |
+----------------+----------------------------+----------------------------------+
SELECT * FROM WritesBook;
+----------+----------------+
| staff_ID | publication_ID |
+----------+----------------+
|     6992 |           1001 |
|     6992 |           1002 |
|     6994 |           1003 |
|     6996 |           1004 |
+----------+----------------+
```

```
SELECT * FROM Articles;
+----------------+-----------------------+------------------------+---------------+
| publication_ID | title                 | text                   | creation_date |
+----------------+-----------------------+------------------------+---------------+
|           1005 | Science Direct Today  | science today text     | 0000-00-00    |
|           1005 | science today title2  | science today text2    | 2022-04-02    |
|           1006 | Health Today          | Health and Fitness     | 0000-00-00    |
|           1006 | health today title    | health today text      | 2022-09-02    |
|           1006 | health today title2   | health today text2     | 2022-12-02    |
|           1007 | fashion today title   | fashion today text     | 2022-11-09    |
|           1007 | fashion today title2  | fashion today text2    | 2022-02-07    |
|           1008 | Animal Science        | Animal Science insights | 0000-00-00    |
|           1008 | technology today title | technology today text  | 2022-02-08    |
+----------------+-----------------------+------------------------+---------------+
```

```
SELECT * FROM WritesArticles;
+----------+----------------+-----------------------+
| staff_ID | publication_ID | title                 |
+----------+----------------+-----------------------+
|     6994 |           1008 | technology today title |
|     6996 |           1006 | health today title    |
|     6996 |           1006 | health today title2   |
|     6996 |           1007 | fashion today title   |
|     6996 |           1007 | fashion today title2  |
|     6998 |           1005 | Science Direct Today  |
|     6998 |           1005 | science today title2  |
+----------+----------------+-----------------------+
```

QUERIES:

Editing and Publishing

1.Enter basic information on a new publication:

```
INSERT INTO `Publications` (`publication_ID`, `title`,`topic`,`type`,
`price`)
VALUES (1001,'Brain Science','Science','Book',23);

Query OK, 1 row affected (0.00 sec)
```

2.Update publication information:

```
UPDATE Publications
SET title = 'Technology Everyday'
WHERE publication_ID = 1008;

Query OK, 1 row affected (0.00 sec)
```

3.Assign editor(s) to publication:

```
INSERT INTO Edits VALUES(6995,1009);

Query OK, 1 row affected (0.01 sec)
```

4.Let each editor view the information on the publications he/she is responsible for:

```
SELECT * FROM Publications
WHERE publication_ID IN (
     SELECT publication_ID
     FROM Edits
     WHERE staff_ID = 6995);
+----------------+----------------+---------+------------+-------+
| publication_ID | title          | topic   | type       | price |
+----------------+----------------+---------+------------+-------+
|           1001 | Brain Science  | Science | Book       | 23.00 |
|           1002 | Animal Fashion | Fashion | Book       | 40.00 |
|           1007 | Fashion Today  | Fashion | Periodical | 44.00 |
|           1009 | New Science    | Science | Book       | 25.00 |
+----------------+----------------+---------+------------+-------+
4 rows in set (0.01 sec)
```

5.Edit table of contents of a publication, by adding/deleting articles (for periodic publications) or chapters/sections (for books)

5.1.Adding Chapters

```
INSERT INTO Chapters VALUES(1001,'Science Direct','Science Today');

Query OK, 1 row affected (0.00 sec)
```

5.2.Deleting Chapters

```
DELETE FROM Chapters
WHERE title='Introduction to Food';

Query OK, 1 row affected (0.01 sec)
```

5.3Adding Articles

```
INSERT INTO Articles VALUES(1008,'Animal Science','Animal Science
insights',2022-01-01);

Query OK, 1 row affected, 1 warning (0.01 sec)
```

5.4.Delete Articles

```
DELETE FROM Articles
WHERE title='technology today title2';

Query OK, 1 row affected (0.00 sec)
```

Production of a book edition or of an issue of a publication.

1.Enter a new book edition or new issue of a publication:

```
INSERT INTO Publications VALUES(1012,'Fitness','Health','Book',65);

Query OK, 1 row affected (0.01 sec)

INSERT INTO Books VALUES(1012,123423,1,2022-02-03);

Query OK, 1 row affected, 1 warning (0.00 sec)
```

2.Update a book edition

```
UPDATE Books set ISBN=125459
WHERE publication_ID=1012;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

3. Delete a book edition

```
DELETE FROM Books WHERE ISBN=125459;

Query OK, 1 row affected (0.00 sec)
```

4. Enter an article:

```
INSERT INTO Articles VALUES(1006, 'Health Today','Health and Fitness',
2022-03-19);

Query OK, 1 row affected, 1 warning (0.00 sec)
```

5.Update Article title

```
UPDATE Articles SET title='Science Direct Today' WHERE
publication_ID=1005 and title='science today title';

Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

6.Update Author's name

```
Update Staff
SET name='Charles'
WHERE staff_ID IN(
     SELECT staff_ID
     FROM Edits NATURAL JOIN Articles
     WHERE Articles.Publication_ID=1008 and
     Articles.title='Animal Science');

Query OK, 1 row affect ed (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

7. Update article/chapter topic

```
UPDATE Publications
SET topic='Science'
WHERE publication_ID=1012;

Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

8. Update article/chapter Date

```
UPDATE Articles
SET creation_date=2022-01-01
WHERE publication_ID=1005 and title='Science Direct Today';

Query OK, 1 row affected, 1 warning (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 1
```

9. Update text of an article

```
UPDATE Articles SET text='Science direct' WHERE publication_ID=1005
and title='Science Direct Today'

Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

10. Find Books by topic

```
SELECT title
FROM Publications
WHERE topic='Health' and type='Book';

+----------------------+
| title                |
+----------------------+
| Introduction to Food |
+----------------------+
1 row in set (0.00 sec)
```

11. Find Articles by topic

```
SELECT
      A.publication_ID,
      P.topic,  A.title
FROM Articles A JOIN Publications P ON A.publication_ID = P.publication_ID
WHERE P.topic='Science';
+----------------+---------+----------------------+
| publication_ID | topic   | title                |
+----------------+---------+----------------------+
|           1005 | Science | Science Direct Today |
|           1005 | Science | science today title2 |
+----------------+---------+----------------------+
```

12. Find Books by date

```
SELECT *
FROM Books
WHERE publication_date='2022-03-02';
+----------------+--------+---------+------------------+
| publication_ID | ISBN   | edition | publication_date |
+----------------+--------+---------+------------------+
|           1003 | 123458 |       1 | 2022-03-02       |
+----------------+--------+---------+------------------+
1 row in set (0.00 sec)
```

13. Find Articles by date

```
SELECT *
FROM Articles
WHERE creation_date='2022-09-02';
+----------------+--------------------+-------------------+---------------+
| publication_ID | title              | text              | creation_date |
+----------------+--------------------+-------------------+---------------+
|           1006 | health today title | health today text | 2022-09-02    |
+----------------+--------------------+-------------------+---------------+
1 row in set (0.01 sec)
```

14. Find Books by author’s name

```
SELECT *
FROM Books
WHERE publication_ID IN(
      select publication_ID
      FROM Staff NATURAL JOIN Edits
      WHERE Staff.name='Harish' AND Staff.role='Author');

+----------------+--------+---------+------------------+
| publication_ID | ISBN   | edition | publication_date |
+----------------+--------+---------+------------------+
|           1001 | 123456 |       1 | 2022-02-02       |
|           1002 | 123457 |       1 | 2022-04-03       |
+----------------+--------+---------+------------------+
2 rows in set (0.00 sec)
```

15. Find articles by author’s name

```
SELECT *
FROM Articles
WHERE publication_ID IN(
      SELECT publication_ID
      FROM Staff NATURAL JOIN Edits
      WHERE Staff.Name='Sandeep' AND Staff.role='Author');
+----------------+---------------------+--------------------+---------------+
| publication_ID | title               | text               | creation_date |
+----------------+---------------------+--------------------+---------------+
|           1006 | Health Today        | Health and Fitness | 0000-00-00    |
|           1006 | health today title  | health today text  | 2022-09-02    |
|           1006 | health today title2 | health today text2 | 2022-12-02    |
+----------------+---------------------+--------------------+---------------+
3 rows in set (0.00 sec)
```

16. Enter payment for author or editor, and keep track of when each payment was claimed by its addressee.

```
INSERT INTO Payments VALUES(6995, 2022-03-02,500, 2022-04-03);
Query OK, 1 row affected, 2 warnings (0.01 sec)

UPDATE Payments SET collection_date='2022-04-04'
WHERE staff_ID=6995 and salary_date=2022-03-02;

Query OK, 1 row affected, 1 warning (0.01 sec)
```

Distribution

1. Enter new distributor

```
     INSERT INTO `Distributors`
     VALUES(3001,9101111111,'Raleigh','111, Avent Ferry
     Road','library','Distributor1',0,'Shane’);

     Query OK, 1 row affected (0.01 sec)
```

2. update distributor information

```
     UPDATE `Distributors`
     SET
          phone = '9951009755',
          city = 'Cary',
          name = 'Distributor9'
     WHERE account_number = 3001;

    Query OK, 1 row affected (0.00 sec)
    Rows matched: 1  Changed: 1  Warnings: 0
```

3. delete a distributor.

```
     DELETE
     FROM Distributors
     WHERE account_number = 30010;
```

```
Query OK, 1 row affected (0.00 sec)
```

4. Input orders from distributors, for a book edition or an issue of a publication per distributor, for a certain date.

```
      INSERT INTO Orders VALUES (13345, 1002, 3001, '2021-03-02',
'2021-03-08', 10, 400, 20);

Query OK, 1 row affected (0.00 sec)
```

5. Bill distributor for an order

```
      UPDATE
      Distributors d JOIN Orders o
      ON o.distributor_account_no = d.account_number
      SET d.balance = d.balance + o.total_cost + o.shipping_cost
      WHERE o.order_number = 13345;

      Query OK, 1 row affected (0.00 sec)
      Rows matched: 1  Changed: 1  Warnings: 0
```

6. change outstanding balance of a distributor on receipt of a payment.

```
      INSERT INTO `DistributorPayments` (`account_number`,
      `payment_date`,`amount_paid`)
      VALUES (3001,'2021-01-02','420');

      Query OK, 1 row affected (0.00 sec)

      UPDATE Distributors d join DistributorPayments dp
      ON dp.account_number = d.account_number
      SET d.balance = d.balance - dp.amount_paid
      WHERE dp.account_number = 3001 AND dp.payment_date =
'2021-01-02';

      Query OK, 1 row affected (0.01 sec)
      Rows matched: 1  Changed: 1  Warnings: 0
```

Reports.

Generate montly reports:

1. number and total price of copies of each publication bought per distributor per month;

```
SELECT
     EXTRACT(YEAR FROM order_date) as year,
     EXTRACT(MONTH FROM order_date) as month,distributor_account_no,
     publication_ID,
     SUM(total_cost) AS order_value,
     SUM(number_of_copies) AS total_copies
FROM Orders
GROUP BY EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM
order_date), distributor_account_no, publication_ID;
```

```
+------+-------+------------------------+----------------+-------------+--------------+
| year | month | distributor_account_no | publication_ID | order_value | total_copies |
+------+-------+------------------------+----------------+-------------+--------------+
| 2021 |     3 |                   3001 |           1002 |      600.00 |           15 |
| 2021 |     3 |                   3001 |           1003 |      240.00 |            5 |
| 2021 |     3 |                   3002 |           1003 |      480.00 |           10 |
| 2021 |     4 |                   3003 |           1004 |      240.00 |           10 |
| 2021 |     5 |                   3002 |           1003 |      960.00 |           20 |
| 2021 |     6 |                   3003 |           1004 |      240.00 |           10 |
+------+-------+------------------------+----------------+-------------+--------------+
```

2.total revenue of the publishing house

```
SELECT
     EXTRACT(YEAR FROM payment_date) AS year,
     EXTRACT(MONTH FROM payment_date) AS month,
     SUM(amount_paid) AS revenue
FROM DistributorPayments
GROUP BY EXTRACT(YEAR FROM payment_date), EXTRACT(MONTH FROM
payment_date);
```

```
+------+-------+---------+
| year | month | revenue |
+------+-------+---------+
| 2021 |     1 | 2000.00 |
| 2021 |     2 |  920.60 |
| 2021 |     6 |  999.00 |
| 2021 |     7 | 1899.00 |
| 2021 |     8 | 1001.00 |
| 2021 |     9 |  987.00 |
+------+-------+---------+
```

3.total expenses (i.e., shipping costs and salaries).

```
     SELECT
          year,
          month,
          SUM(expenses) AS expenses
     FROM(
          SELECT
               EXTRACT(YEAR FROM salary_date) AS year,
               EXTRACT(MONTH FROM salary_date) AS month,
               SUM(payment_amount) AS expenses
          FROM Payments
          GROUP BY EXTRACT(YEAR FROM salary_date),
               EXTRACT(MONTH FROM salary_date)
          UNION
          SELECT
               EXTRACT(YEAR FROM order_delivery_date) AS year,
               EXTRACT(MONTH FROM order_delivery_date) AS month,
               SUM(shipping_cost) AS expenses
          FROM Orders
          GROUP BY EXTRACT(YEAR FROM order_delivery_date),
               EXTRACT(MONTH FROM order_delivery_date)
          ) AS Expenses
     GROUP BY year, month
```

```
+------+-------+----------+
| year | month | expenses |
+------+-------+----------+
| 2021 |     3 |  3040.08 |
| 2021 |     4 |  2018.08 |
| 2021 |     5 |    60.00 |
| 2021 |     6 |    20.00 |
+------+-------+----------+
```

4.Calculate the total current number of distributors

```
SELECT COUNT(account_number) AS total_distributors
FROM Distributors;

+--------------------+
| total_distributors |
+--------------------+
|                  8 |
+--------------------+
```

5.calculate total revenue (since inception) per city

```
SELECT city AS distributor_city, sum(amount_paid) as revenue
FROM Distributors D NATURAL JOIN DistributorPayments DP
GROUP BY city;

+------------------+---------+
| distributor_city | revenue |
+------------------+---------+
| Austin           | 1986.00 |
| Boston           |  790.00 |
| Cary             | 1210.00 |
| Chicago          | 2697.60 |
| New York         |  122.00 |
| Raleigh          | 1001.00 |
+------------------+---------+
```

6.calculate total revenue (since inception) per distributor.

```
     SELECT
          account_number AS distributor_account_no,
          sum(amount_paid) as revenue
     FROM DistributorPayments
     GROUP BY account_number;
```

```
+------------------------+---------+
| distributor_account_no | revenue |
+------------------------+---------+
|                   3001 | 1210.00 |
|                   3002 |  122.00 |
|                   3003 | 1899.00 |
|                   3004 |  790.00 |
|                   3005 |  987.00 |
|                   3006 | 1001.00 |
|                   3007 |  798.60 |
|                   3008 |  999.00 |
+------------------------+---------+
```

7.calculate total revenue (since inception) per location.

```
     SELECT
          city,
          street_address,
          sum(amount_paid) as revenue
     FROM Distributors D NATURAL JOIN DistributorPayments DP
     GROUP BY city, street_address;
```

```
+----------+------------------------+---------+
| city     | street_address         | revenue |
+----------+------------------------+---------+
| Austin   | 555, Second Street     |  987.00 |
| Austin   | 888, Third Street      |  999.00 |
| Boston   | 444, Silver Spear Lane |  790.00 |
| Cary     | 111, Avent Ferry Road  | 1210.00 |
| Chicago  | 333, High Point Ave    | 1899.00 |
| Chicago  | 777, North 6th Lane    |  798.60 |
| New York | 222, Brooke Street     |  122.00 |
| Raleigh  | 666, Walnutwood        | 1001.00 |
+----------+------------------------+---------+
```

8.Calculate total payments to the editors and authors, per time period and per work type (book authorship, article authorship, or editorial work).

```
SELECT
     S.role AS staff_role,
     A.type AS staff_type,
     SUM(payment_amount) AS total_payments
FROM Staff S NATURAL JOIN Authors A
     NATURAL JOIN Payments SP
WHERE salary_date >= '2021-01-01' AND
     salary_date <= '2021-12-31'
GROUP BY S.role , A.type
UNION
SELECT
     S.role AS staff_role,
     E.type AS staff_type,
     SUM(payment_amount) AS total_payments
FROM Staff S NATURAL JOIN Editors E
     NATURAL JOIN Payments SP
WHERE salary_date >= '2021-01-01' AND
     salary_date <= '2021-12-31'
GROUP BY S.role , E.type;
```

```
+------------+------------+----------------+
| staff_role | staff_type | total_payments |
+------------+------------+----------------+
| Author     | Invited    |        1200.04 |
| Author     | Staff      |        3600.08 |
| Editor     | Invited    |          89.02 |
| Editor     | Staff      |          89.02 |
+------------+------------+----------------+
```

4.2 Analyzing SQL Queries using EXPLAIN directive and optimizing using Indexes

1. Return Books by topic

```
EXPLAIN
SELECT title
FROM Publications
WHERE topic='Health' and type='Book';
```

```
+------+-------------+--------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table        | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+--------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Publications | ALL  | NULL          | NULL | NULL    | NULL |   10 | Using where |
+------+-------------+--------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Here the query is using the where clause and it is doing a full table scan of the table Publications to search for all the rows which have topic = ‘health’

We can optimize this by adding an index to the topic, type columns, which will avoid a table scan.

```
CREATE INDEX TopicTypeIndex on Publications(topic,type);
```

```
EXPLAIN
SELECT title
FROM Publications
WHERE topic='Health' and type='Book';
```

```
+------+-------------+--------------+------+----------------+----------------+---------+-------------+------+-----------------------+
| id   | select_type | table        | type | possible_keys  | key            | key_len | ref         | rows | Extra                 |
+------+-------------+--------------+------+----------------+----------------+---------+-------------+------+-----------------------+
|    1 | SIMPLE      | Publications | ref  | TopicTypeIndex | TopicTypeIndex | 64      | const,const |    1 | Using index condition |
+------+-------------+--------------+------+----------------+----------------+---------+-------------+------+-----------------------+
1 row in set (0.00 sec)
```

With the index added to the table, we now see that we are using the index to retrieve the entries in the table.

2. Return books by publication_date

```
EXPLAIN SELECT *
FROM Books
WHERE publication_date='2022-03-02';
```

```
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Books | ALL  | NULL          | NULL | NULL    | NULL |    4 | Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Here the query is using the where clause and it is doing a full table scan of the table Books to search for all the rows which have publication_date='2022-03-02'.

We can optimize this by adding an index to the column publication_date which will avoid a full table scan.

```
CREATE INDEX publicationDateIndex on Books(publication_date);
```

```
EXPLAIN
SELECT *
FROM Books
WHERE publication_date='2022-03-02';
```

```
+------+-------------+-------+------+----------------------+----------------------+---------+-------+------+-------+
| id   | select_type | table | type | possible_keys        | key                  | key_len | ref   | rows | Extra |
+------+-------------+-------+------+----------------------+----------------------+---------+-------+------+-------+
|    1 | SIMPLE      | Books | ref  | publicationDateIndex | publicationDateIndex | 3       | const |    1 |       |
+------+-------------+-------+------+----------------------+----------------------+---------+-------+------+-------+
```

With the index added to the table, we now see that we are using the index publicationDateIndex to retrieve the entries in the table.

4.3) Evaluating Correctness of SQL Queries with Relational Algebra

1.) Calculate total revenue (since inception) per city.

```
SELECT
      city AS distributor_city,
      sum(amount_paid) as revenue
FROM Distributors d NATURAL JOIN DistributorPayments dp
GROUP BY city;
```

$$\begin{array}{l}\pi_{\text{city} \rightarrow \text{distributor\_city},\ \text{SUM (amount\_paid)} \rightarrow \text{revenue}} \\ \quad (\gamma_{\text{city},\ \text{SUM (amount\_paid)}} \\ \quad\quad (\rho_{d}\ \text{distributors} \bowtie \rho_{dp}\ \text{distributorpayments}))\end{array}$$

Correctness of the query:
Suppose *d* is any tuple in the Distributors relation, and *dp* is any tuple in the DistributorPayments relation, such that the value *d.account_number* is the same as the value *dp.account_number*. Each such combination of tuples *(d, dp)*gives information about one distributor, together with all information on the DistributorPayments. This will contain information about all the payments made by the distributor. For each such combination *(d, dp)*, the query returns the value of distributor_city and revenue. These values are the city of the distributor and the sum(amount_paid) of the payments made by the distributor from distributorpayments. But this is exactly what our query should return.

2.) Find Articles by topic

```
SELECT
      A.publication_ID,
      P.topic,  A.title
FROM Articles A JOIN Publications P ON A.publication_ID = P.publication_ID
WHERE P.topic='Science';
```

$$\begin{array}{l}\pi_{\text{a.publication\_id},\ \text{p.topic},\ \text{a.title}} \\ \quad (\sigma_{\text{p.topic = "Science"}} \\ \quad\quad (\rho_{a}\ \text{articles} \bowtie_{\text{a.publication\_ID = p.publication\_ID}} \rho_{p}\ \text{publications}))\end{array}$$

Suppose *a* is any tuple in the Articles relation, and *p* is any tuple in the Publications relation, such that the value *a.publication_ID* is the same as the value *p.publication_ID* (The natural join of Articles and Publications gave an erroneous result as the natural join took the title as the criteria to join. To fix this, a theta join had to be performed with a.publication_ID = p.publication_ID). Each such combination of tuples *(a,p)*gives information about one article, together with all information about its publication. For each such combination *(a,p)*, the query returns the value of publication_ID, topic, and title. These values are the publication_ID of the publication, the topic of the publication, and the title of the article. But this is exactly what our query should return.