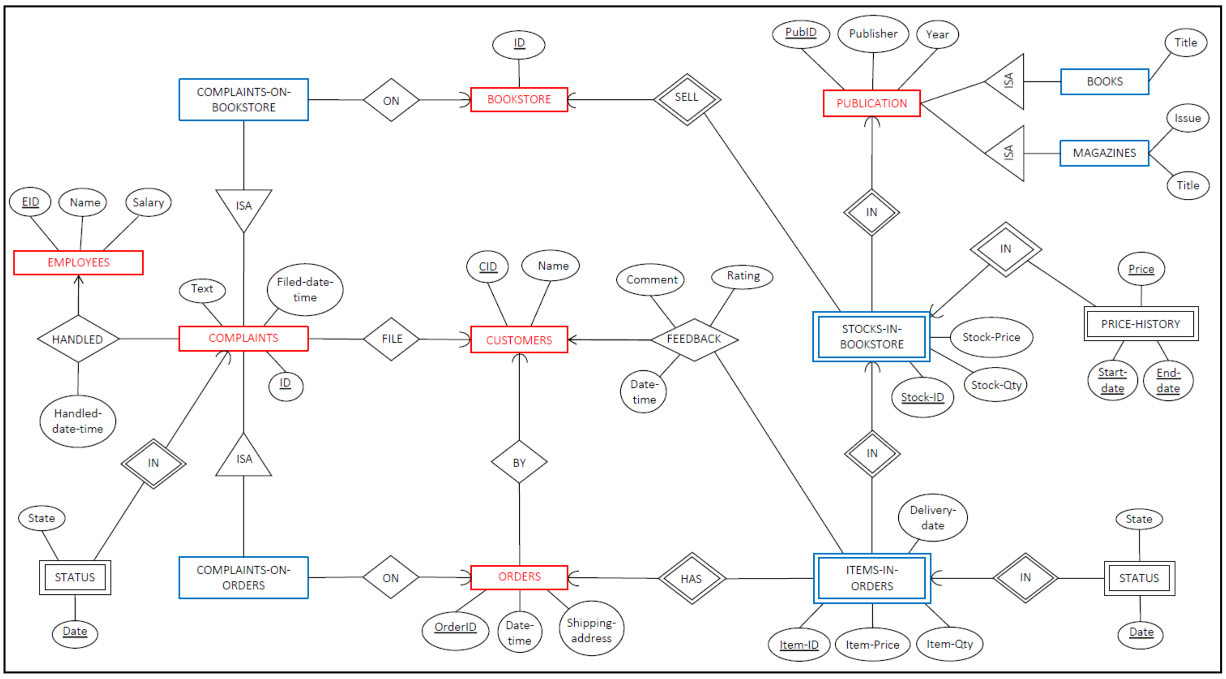
<u>Lab 5 Submission - Team 2 (Z43)</u>

Team members:

- Joel Tan (U2122877C)
- Tar Sreeja (U2123104B)
- Liu Liwen (N2202357F)
- Agarwal Lakshya (U2123904L)
- Luo Minjuan (N2202356J)
- Shourya Kuchhal (U2123334A)

ER Diagram Referenced:



CREATING TABLES

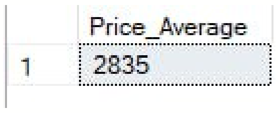
```
CREATE TABLE PUBLICATION (
        PubID INT NOT NULL PRIMARY KEY,
        Publisher VARCHAR(255),
        Year INT
);

CREATE TABLE BOOKS (
   PubID INT PRIMARY KEY,
   Title VARCHAR(255),
   FOREIGN KEY (PubID) REFERENCES PUBLICATION(PubID)
);

CREATE TABLE MAGAZINES (
   PubID INT PRIMARY KEY,
   Title VARCHAR(255),
   Issue INT,
   FOREIGN KEY (PubID) REFERENCES PUBLICATION(PubID)
);

CREATE TABLE BOOKSTORE (
        Bookstore_ID INT NOT NULL PRIMARY KEY
);

CREATE TABLE STOCKS_IN_BOOKSTORE (
        Bookstore_ID INT NOT NULL,
        PubID INT NOT NULL,
        Stock_ID VARCHAR(255) NOT NULL,
        Stock_Price FLOAT,
        Stock_Qty INT,
        PRIMARY KEY(Bookstore_ID, PubID, Stock_ID),
        FOREIGN KEY (Bookstore_ID) REFERENCES BOOKSTORE(Bookstore_ID),
        FOREIGN KEY (PubID) REFERENCES PUBLICATION(PubID)
);

CREATE TABLE PRICE_HISTORY (
   PubID INT NOT NULL,
   Bookstore_ID INT NOT NULL,
   Stock_ID VARCHAR(255) NOT NULL,
   Price FLOAT NOT NULL,
   Start_date DATETIME NOT NULL,
   End_date DATETIME NOT NULL,
   PRIMARY KEY (PubID, Bookstore_ID, Stock_ID, Start_date, End_date),
   FOREIGN KEY (Bookstore_ID, PubID, Stock_ID) REFERENCES
STOCKS_IN_BOOKSTORE(Bookstore_ID, PubID, Stock_ID)
);

CREATE TABLE CUSTOMERS (
        CID INT NOT NULL PRIMARY KEY,
        Name VARCHAR(255)
);

CREATE TABLE ORDERS (
  OrderID INT NOT NULL PRIMARY KEY,
  Order_Date_Time DATETIME,
  Shipping_address VARCHAR(255),
  CID INT
);

CREATE TABLE ITEMS_IN_ORDERS (
  PubID INT NOT NULL,
  Bookstore_ID INT NOT NULL,
  Stock_ID VARCHAR(255) NOT NULL,
  OrderID INT NOT NULL,
  Item_ID INT NOT NULL,
  Item_Price FLOAT,
  Item_Qty INT,
  Delivery_Date DATETIME,
  CID INT,
  Comment VARCHAR(255),
  Rating INT,
  Feedback_Date_Time DATETIME,
  PRIMARY KEY (PubID, Bookstore_ID, Stock_ID, OrderID, Item_ID),
  FOREIGN KEY (PubID, Bookstore_ID, Stock_ID) REFERENCES
STOCKS_IN_BOOKSTORE(PubID, Bookstore_ID, Stock_ID),
  FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID),
  FOREIGN KEY (CID) REFERENCES CUSTOMERS(CID)
);

CREATE TABLE ORDER_STATUS (
        PubID INT NOT NULL,
        Bookstore_ID INT NOT NULL,
        Stock_ID VARCHAR(255) NOT NULL,
        OrderID INT NOT NULL,
        Item_ID INT NOT NULL,
        Date DATETIME NOT NULL,
        State VARCHAR(255),
        PRIMARY KEY (PubID, Bookstore_ID, Stock_ID, OrderID, Item_ID, Date),
        FOREIGN KEY (PubID, Bookstore_ID, Stock_ID, OrderID, Item_ID)
REFERENCES ITEMS_IN_ORDERS
);

CREATE TABLE EMPLOYEES (
        EID INT PRIMARY KEY,
        Name VARCHAR(255),
        Salary INT
);

CREATE TABLE COMPLAINTS (
        Complaint_ID INT PRIMARY KEY,
        Text VARCHAR(255),
        Filed_date_time DATETIME,
        EID INT,
        Handled_date_time DATETIME,
        CID INT,
        FOREIGN KEY(EID) REFERENCES EMPLOYEE(EID),
        FOREIGN KEY(CID) REFERENCES CUSTOMERS(CID),
);

CREATE TABLE COMPLAINT_STATUS (
   Complaint_ID INT NOT NULL,
   Date DATETIME NOT NULL,
   State VARCHAR(255),
   PRIMARY KEY (Complaint_ID, Date),
   FOREIGN KEY (Complaint_ID) REFERENCES COMPLAINTS(Complaint_ID)
);

CREATE TABLE COMPLAINTS_ON_BOOKSTORE (
        Complaint_ID INT NOT NULL PRIMARY KEY,
        Bookstore_ID INT
        FOREIGN KEY (Complaint_ID) REFERENCES COMPLAINTS(Complaint_ID)
        FOREIGN KEY (Bookstore_ID) REFERENCES BOOKSTORE(Bookstore_ID)
);

CREATE TABLE COMPLAINTS_ON_ORDER (
        Complaint_ID INT NOT NULL PRIMARY KEY,
        Order_ID INT,
        FOREIGN KEY (Complaint_ID) REFERENCES COMPLAINTS(Complaint_ID)
        FOREIGN KEY (Order_ID) REFERENCES ORDERS(Order_ID)
);
```

QUERIES

1. Find the average price of “Harry Potter Finale" on Ahamazon from 1 August 2022 to 31 August 2022.

SELECT AVG(Price) as Price_Average
FROM BOOKS X, PRICE_HISTORY Y
WHERE (Y.Start_date <= ‘2022-08-01 00:00:00.000’ AND Y.End_date >= ‘2022-08-31 00:00:00.000’) AND X.Title = ‘Harry Potter Finale’ AND X.PubID = Y.PubID;

Query Output



	Price_Average
1	2835

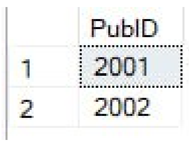
Explanation

In this query, we join the BOOKS and PRICE_HISTORY tables on the condition that their PubID attributes match. We then select the records where the Start_date and End_date are between 1 August 2022 and 31 August 2022, and the title of the book is ‘Harry Potter Finale’. We calculate the average price of the book by using the AVG() function on the Price column in the joined tables.

2. Find publications that received at least 10 ratings of “5” in August 2022, and rank them by their average ratings.

SELECT PubID
FROM
(SELECT X.PubID, COUNT(*) AS RATINGCOUNT, AVG(Rating) AS AVGRATING
FROM PUBLICATION X, ITEMS_IN_ORDERS Y
WHERE Y.Rating = 5 AND MONTH(Y.Feedback_Date_Time) = 8 AND YEAR(Feedback_Date_Time) = 2022 AND X.PubID = Y.PubID
GROUP BY X.PubID) AS Z
WHERE RATINGCOUNT >= 10
ORDER BY AVGRATING DESC;

Query Output



	PubID
1	2001
2	2002

Explanation

This SQL query finds the PubID of the publications that received 10 or more ratings with a rating of 5 in the month of August 2022. The result is sorted by the average rating of the publication in descending order.

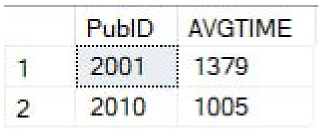
In this query, we first join the PUBLICATION and ITEMS_IN_ORDERS tables such that their PubID attributes match. We then filter the joined table by selecting only the tuples with a rating of 5 and feedback date within the month of August 2022. Then for each publication (GROUP BY PubID), we calculate the number of ratings that are equal to 5 and the average rating.

We further filter the result of this intermediate table using a subquery to select only the publications that have at least 10 ratings of 5 (RATINGCOUNT >= 10). Finally, we select the publication ID from this filtered table and sort the result by the average rating in descending order.

3. For all publications purchased in June 2022 that have been delivered, find the average time from the ordering date to the delivery date.

**SELECT X.PubID, AVG (DATEDIFF(day, Y.Delivery_Date, Z.Order_Date_Time)) AS AVGTIME
FROM PUBLICATION X, ITEMS_IN_ORDERS Y, ORDERS Z, ORDER_STATUS S
WHERE X.PubID = Y.PubID AND Y.OrderID = Z.OrderID AND Z.OrderID = S.OrderID AND Y.Delivery_Date IS NOT NULL AND MONTH(Order_Date_Time) = 6 AND YEAR(Order_Date_Time) = 2022 and S.State = ‘delivered’
GROUP BY X.PubID;**

Query Output



	PubID	AVGTIME
1	2001	1379
2	2010	1005

Explanation

In this query, we join four tables: PUBLICATION, ITEMS_IN_ORDERS, ORDERS, and ORDER_STATUS, based on the publication ID and order ID attributes. We then select only orders that have been delivered, which is determined by checking if the delivery date is not null and if the order status is ‘delivered’. We also select only those orders that have been placed in June 2022.

We then apply the DATEDIFF function to calculate the difference in days between the delivery date and the order date for each item in an order. Finally, we group these differences by the publication ID and apply the AVG function to compute the average delivery time for each publication.

4. Let us define the “latency” of an employee by the average time that he/she takes to process a complaint.
Find the employee with the smallest latency.

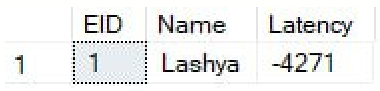
View created:

```
CREATE VIEW LATENCIES AS
SELECT AVG(DATEDIFF(day, Handled_date_time, Filed_date_time)) AS Latency, C.EID
AS EID, Name
FROM COMPLAINTS C, EMPLOYEES E
WHERE C.EID = E.EID
GROUP BY C.EID, Name;
```

Query:

```
SELECT EID, Name, Latency
FROM LATENCIES,
(SELECT MIN (LATENCY) AS MinLatency
FROM LATENCIES) AS Z
WHERE Latency = MinLatency;
```

Query Output



	EID	Name	Latency
1	1	Lashya	-4271

Explanation

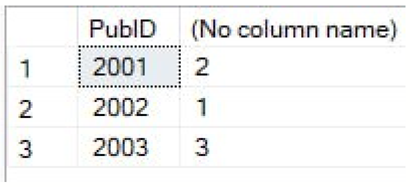
Using the first query, we create a view called "Latencies" that contains the latency of each employee, where latency is the difference between the time a complaint is filed and the time it is handled. The query joins the "COMPLAINTS" and "EMPLOYEES" tables on the employee ID (EID) and groups the results by EID and Name.

In the second query, we select the EID, Name, and Latency columns from the Latencies view and filter the results to only show the records where the latency is equal to the minimum latency calculated in the subquery (SELECT MIN(LATENCY) AS MinLatency FROM LATENCIES). This effectively identifies the employee(s) with the smallest latency.

5. Produce a list that contains (i) all publications published by Nanyang Publisher Company, and (ii) for
each of them, the number of bookstores on Ahamazon that sell them.

```
SELECT P.PubID, COUNT (S.Bookstore_ID)
FROM PUBLICATION P, STOCKS_IN_BOOKSTORE S
WHERE P.PubID = S.PubID AND P.Publisher = 'Nanyang Publisher Company'
GROUP BY P.PubID;
```

__Query Output__



	PubID	(No column name)
1	2001	2
2	2002	1
3	2003	3

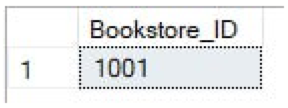
__Explanation__

In this query, we join two tables, PUBLICATIONS and STOCKS_IN_BOOKSTORE, on the PubID column and select the publications where the publisher is "Nanyang Publisher Company". We then group the results by the publication ID attribute, and count the number of rows in the STOCKS_IN_BOOKSTORE table for each publication. Thus, the resulting output contains two columns, the PubIDs of the publications published by Nanyang Publisher Company, and for each publication, the number of bookstores that sell them.

6. Find bookstores that made the most revenue in August 2022.

```
SELECT Bookstore_ID
FROM
(SELECT Bookstore_ID, SUM(Item_Price) AS PriceSum
FROM ITEMS_IN_ORDERS X, ORDERS Y
WHERE X.OrderID = Y.OrderID AND MONTH(Y.Order_Date_Time) = 8 AND
YEAR(Y.Order_Date_Time) = 2022
GROUP BY X.Bookstore_ID) AS Z
WHERE Z.PriceSum =
(SELECT MAX(PriceSum)
FROM
(SELECT SUM(Item_Price) AS PriceSum
FROM ITEMS_IN_ORDERS X, ORDERS Y
WHERE X.OrderID = Y.OrderID AND MONTH(Y.Order_Date_Time) = 8 AND
YEAR(Y.Order_Date_Time) = 2022
GROUP BY X.Bookstore_ID) AS Z
);
```

Query Output



	Bookstore_ID
1	1001

Explanation

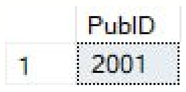
Using this subquery, we first join the ITEMS_IN_ORDERS and ORDERS tables using their common attribute, OrderID. We then group the results by Bookstore_ID and calculate the total sales amount for each Bookstore_ID using the SUM() function.

In the outer query, we select the Bookstore_ID(s) that have the highest total sales, by comparing the total sales amount for each Bookstore_ID with the maximum sales amount, which is obtained by a subquery that selects the maximum value of PriceSum from the previous subquery. Any Bookstore_ID(s) with a total sales amount equal to the maximum sales amount, i.e., the maximum revenue will be returned.

7. For customers that made the most number of complaints, find the most expensive publication he/she has ever purchased.

```
SELECT TOP 1 PubID
FROM (
SELECT I.PubID, MAX(H.Price) AS Max_Price
FROM COMPLAINTS C
JOIN ORDERS O ON C.CID = O.CID
JOIN ITEMS_IN_ORDERS I ON O.OrderID = I.OrderID
JOIN PRICE_HISTORY H ON I.PubID = H.PubID AND I.Bookstore_ID = H.Bookstore_ID
AND I.Stock_ID = H.Stock_ID
JOIN (
   SELECT TOP 1 CID
   FROM COMPLAINTS
   GROUP BY CID
   ORDER BY COUNT(*) DESC
) AS Max_Cust ON C.CID = Max_Cust.CID
GROUP BY I.PubID) AS Z
ORDER BY Max_Price DESC;
```

Query Output



	PubID
1	2001

Explanation

In this query, we select the most expensive publication purchased by the customer who has made the most number of complaints.

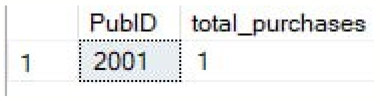
Using the subquery, we select the publication ID attribute and its maximum price from the items in orders that have been placed by the customer who has made the maximum number of complaints. This is done by joining the COMPLAINTS, ORDERS, ITEMS_IN_ORDERS, and PRICE_HISTORY tables together and filtering by the customer with the most complaints. The results are grouped by publication ID.

The outer query selects the top 1 row from the subquery, which corresponds to the publication with the highest maximum price, i.e., the most expensive publication among all publications purchased by the customer with the most complaints.

8. Find publications that have never been purchased by any customer in July 2022, but are the top 3 most purchased publications in August 2022.

```
SELECT TOP 3 PUBLICATION.PubID, COUNT(*) AS total_purchases
FROM ITEMS_IN_ORDERS
JOIN PUBLICATION ON PUBLICATION.PubID = ITEMS_IN_ORDERS.PubID
JOIN ORDERS ON ORDERS.OrderID = ITEMS_IN_ORDERS.OrderID
WHERE Order_Date_Time >= '2022-08-01' AND Order_Date_Time <= '2022-08-31'
   AND PUBLICATION.PubID NOT IN (
         SELECT DISTINCT ITEMS_IN_ORDERS.PubID
         FROM ITEMS_IN_ORDERS
         WHERE Order_Date_Time >= '2022-07-01' AND Order_Date_Time <=
'2022-07-31'
   )
GROUP BY PUBLICATION.PubID
ORDER BY total_purchases DESC;
```

Query Output



```
     PubID   total_purchases
1    2001    1
```

Explanation

In this query, we join two tables, ITEMS_IN_ORDERS and PUBLICATION, on their common PubID attribute. We then filter the results to only include rows where the Delivery_date falls in the month of August 2022.

We also include a "NOT IN" clause that filters out any publications that were also purchased during July 2022. This is done by running a subquery on ITEMS_IN_ORDERS table that retrieves all distinct PubIDs with a Delivery_date that falls between July 1st and July 31st, and then excluding those PubIDs from the main query.

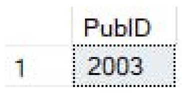
We then group the results by PubID, and calculate the total number of purchases for each publication during August 2022 using the COUNT function.

Finally, we order the results by the total number of purchases in descending order and retrieve only the top 3 publications that were purchased during August 2022, but never purchased in July 2022.

9. Find publications that are increasingly being purchased over at least 3 months.

```
SELECT DISTINCT X.PubID
FROM
(SELECT PubID, SUM(Item_Qty) AS ICount, Order_Date_Time FROM
ITEMS_IN_ORDERS I1 JOIN ORDERS O1 ON I1.OrderID = O1.OrderID
GROUP BY I1.PubID, O1.Order_Date_Time) AS X
JOIN
(SELECT PubID, SUM(Item_Qty) AS ICount, Order_Date_Time FROM
ITEMS_IN_ORDERS I2 JOIN ORDERS O2 ON I2.OrderID = O2.OrderID
GROUP BY I2.PubID, O2.Order_Date_Time) AS Y
ON X.PubID = Y.PubID AND MONTH(Y.Order_Date_Time) =
MONTH(X.Order_Date_Time) + 1
AND YEAR(Y.Order_Date_Time) = YEAR(X.Order_Date_Time)
JOIN
(SELECT PubID, SUM(Item_Qty) AS ICount, Order_Date_Time FROM
ITEMS_IN_ORDERS I3 JOIN ORDERS O3 ON I3.OrderID = O3.OrderID
GROUP BY I3.PubID, O3.Order_Date_Time) AS Z
ON Y.PubID = Z.PubID AND MONTH(Z.Order_Date_Time) =
MONTH(Y.Order_Date_Time) + 1
                                        AND YEAR(Z.Order_Date_Time) =
YEAR(Y.Order_Date_Time)
WHERE X.ICount < Y.ICount AND Y.ICount < Z.ICount;
```

<u>**Query Output**</u>



	PubID
1	2003

<u>**Explanation**</u>

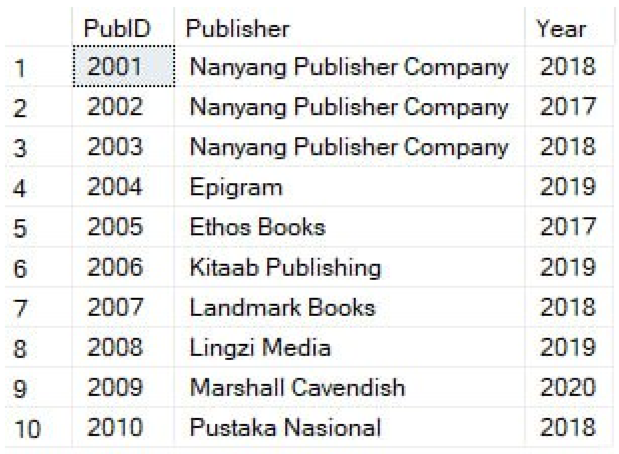
In this query, we find publication IDs of the publications for which the number of purchases increases for 3 consecutive months.

We first create two subqueries (X and Y) that calculate the total item quantity sold per month for each publication. We then join the two subqueries with another subquery (Z) that calculates the total item quantity sold per month for each publication, in the month following Y's month. The join is performed on the PubID attribute and checks that Y's month is one month prior to Z's month.

Finally, we filter the results to only include PubIDs of publications for which the number of purchases in the first month (X) is less than the number of purchases in the second month (Y), and the number of purchases in the second month (Y) is less than the number of purchases in the third month (Z), i.e., the publication is increasingly purchased over the 3 months. The DISTINCT keyword is used to remove any duplicate PubIDs that may result from the join.

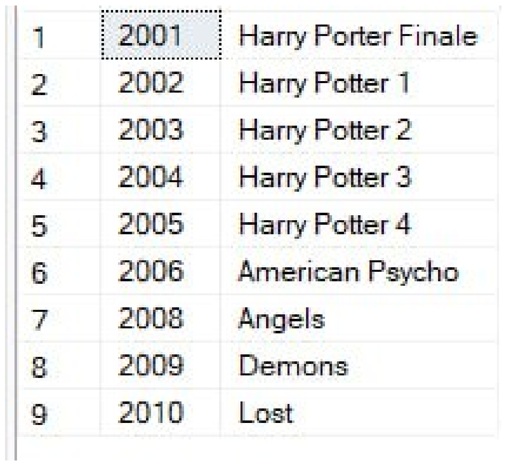
RECORDS

PUBLICATION



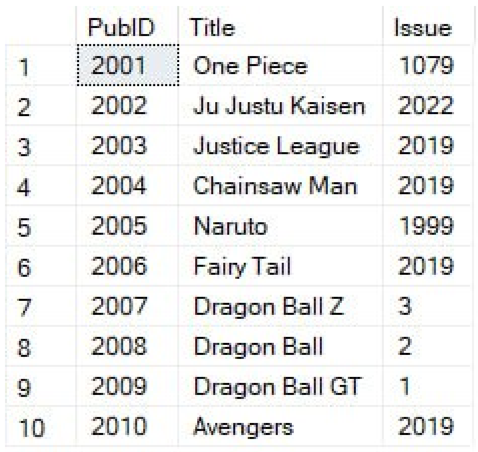
	PubID	Publisher	Year
1	2001	Nanyang Publisher Company	2018
2	2002	Nanyang Publisher Company	2017
3	2003	Nanyang Publisher Company	2018
4	2004	Epigram	2019
5	2005	Ethos Books	2017
6	2006	Kitaab Publishing	2019
7	2007	Landmark Books	2018
8	2008	Lingzi Media	2019
9	2009	Marshall Cavendish	2020
10	2010	Pustaka Nasional	2018

BOOKS



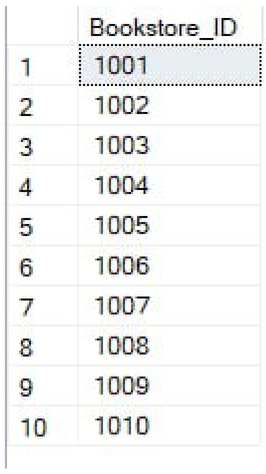
1	2001	Harry Porter Finale
2	2002	Harry Potter 1
3	2003	Harry Potter 2
4	2004	Harry Potter 3
5	2005	Harry Potter 4
6	2006	American Psycho
7	2008	Angels
8	2009	Demons
9	2010	Lost

MAGAZINES



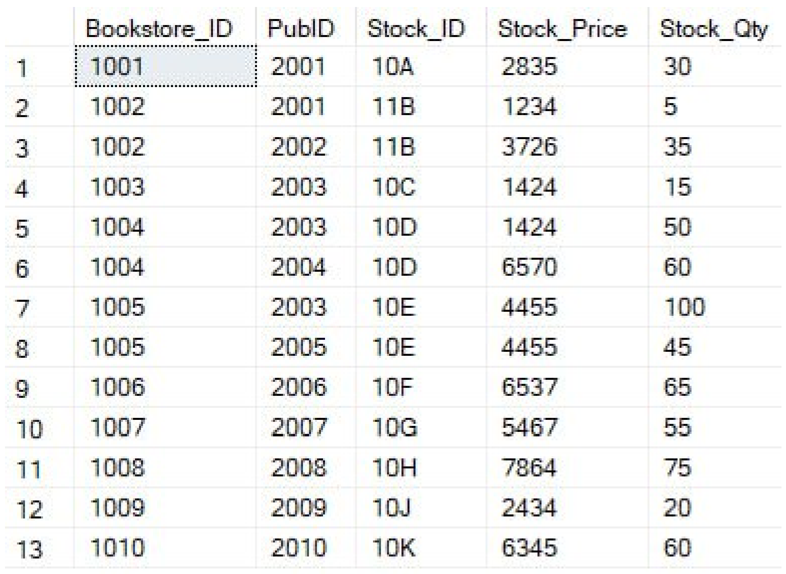
	PubID	Title	Issue
1	2001	One Piece	1079
2	2002	Ju Justu Kaisen	2022
3	2003	Justice League	2019
4	2004	Chainsaw Man	2019
5	2005	Naruto	1999
6	2006	Fairy Tail	2019
7	2007	Dragon Ball Z	3
8	2008	Dragon Ball	2
9	2009	Dragon Ball GT	1
10	2010	Avengers	2019

BOOKSTORE



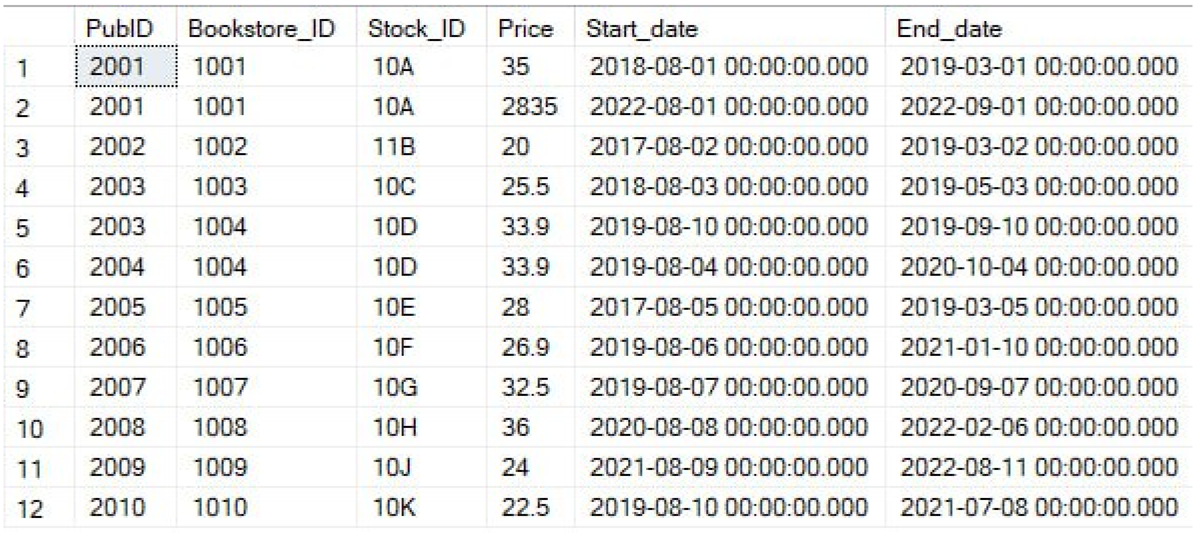
	Bookstore_ID
1	1001
2	1002
3	1003
4	1004
5	1005
6	1006
7	1007
8	1008
9	1009
10	1010

STOCKS_IN_BOOKSTORE



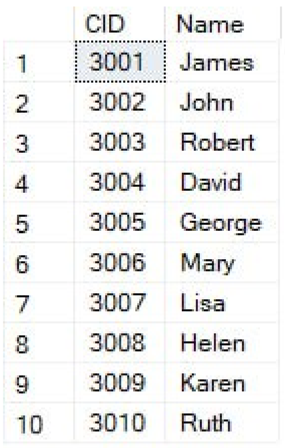
	Bookstore_ID	PubID	Stock_ID	Stock_Price	Stock_Qty
1	1001	2001	10A	2835	30
2	1002	2001	11B	1234	5
3	1002	2002	11B	3726	35
4	1003	2003	10C	1424	15
5	1004	2003	10D	1424	50
6	1004	2004	10D	6570	60
7	1005	2003	10E	4455	100
8	1005	2005	10E	4455	45
9	1006	2006	10F	6537	65
10	1007	2007	10G	5467	55
11	1008	2008	10H	7864	75
12	1009	2009	10J	2434	20
13	1010	2010	10K	6345	60

PRICE_HISTORY



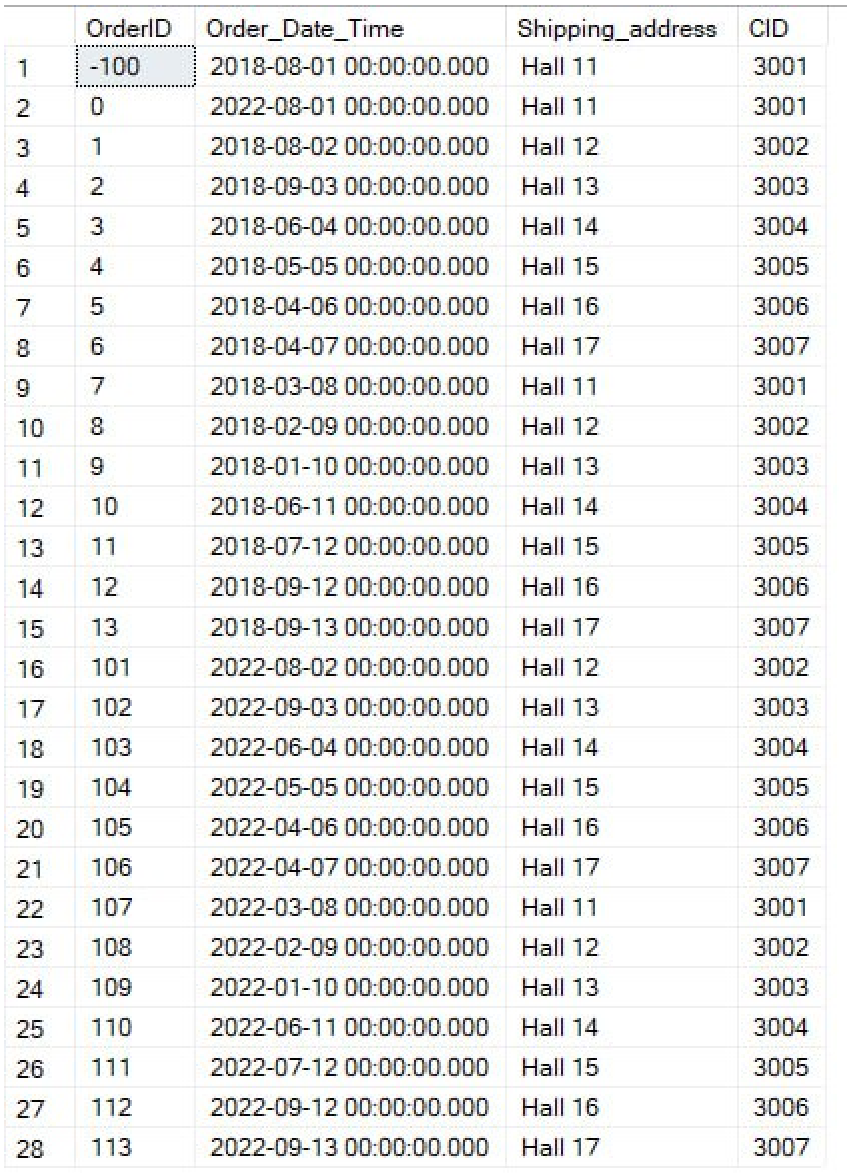
	PubID	Bookstore_ID	Stock_ID	Price	Start_date	End_date
1	2001	1001	10A	35	2018-08-01 00:00:00.000	2019-03-01 00:00:00.000
2	2001	1001	10A	2835	2022-08-01 00:00:00.000	2022-09-01 00:00:00.000
3	2002	1002	11B	20	2017-08-02 00:00:00.000	2019-03-02 00:00:00.000
4	2003	1003	10C	25.5	2018-08-03 00:00:00.000	2019-05-03 00:00:00.000
5	2003	1004	10D	33.9	2019-08-10 00:00:00.000	2019-09-10 00:00:00.000
6	2004	1004	10D	33.9	2019-08-04 00:00:00.000	2020-10-04 00:00:00.000
7	2005	1005	10E	28	2017-08-05 00:00:00.000	2019-03-05 00:00:00.000
8	2006	1006	10F	26.9	2019-08-06 00:00:00.000	2021-01-10 00:00:00.000
9	2007	1007	10G	32.5	2019-08-07 00:00:00.000	2020-09-07 00:00:00.000
10	2008	1008	10H	36	2020-08-08 00:00:00.000	2022-02-06 00:00:00.000
11	2009	1009	10J	24	2021-08-09 00:00:00.000	2022-08-11 00:00:00.000
12	2010	1010	10K	22.5	2019-08-10 00:00:00.000	2021-07-08 00:00:00.000

CUSTOMERS



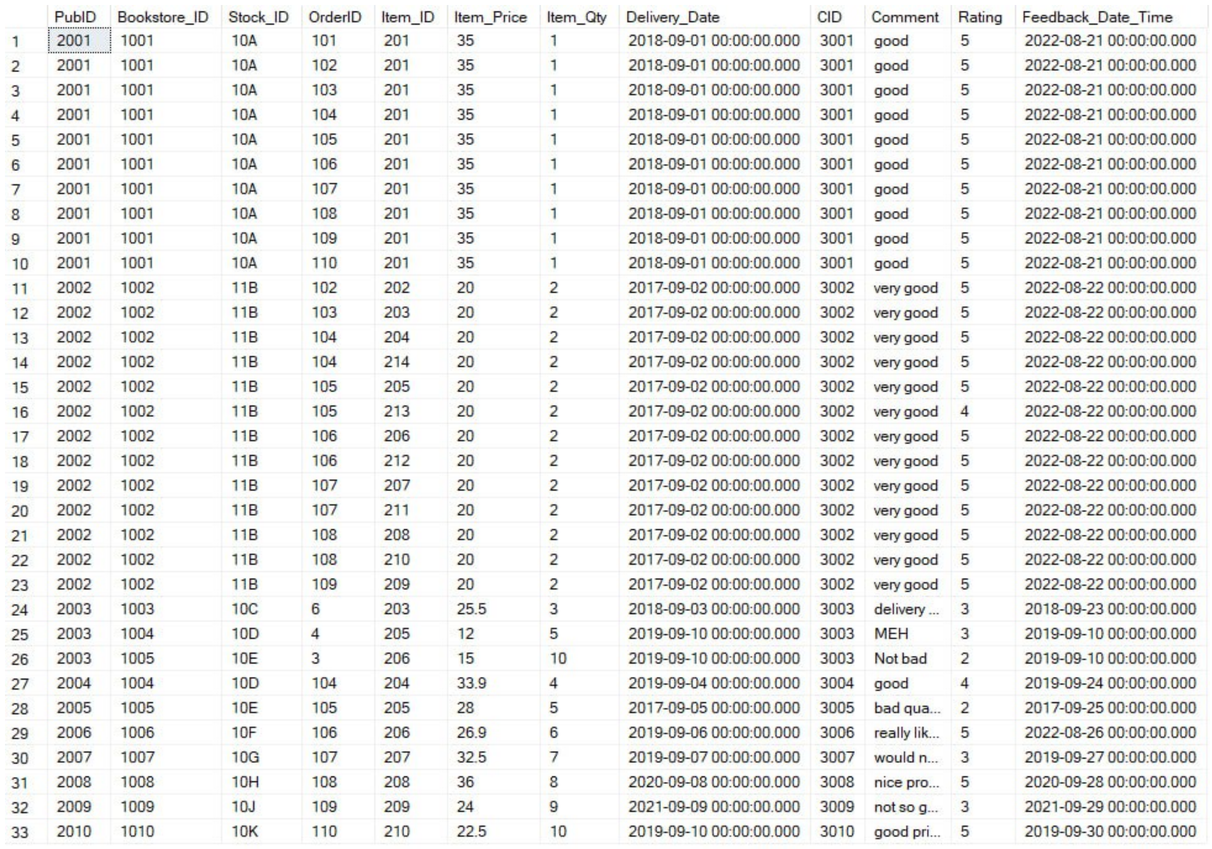
	CID	Name
1	3001	James
2	3002	John
3	3003	Robert
4	3004	David
5	3005	George
6	3006	Mary
7	3007	Lisa
8	3008	Helen
9	3009	Karen
10	3010	Ruth

ORDERS



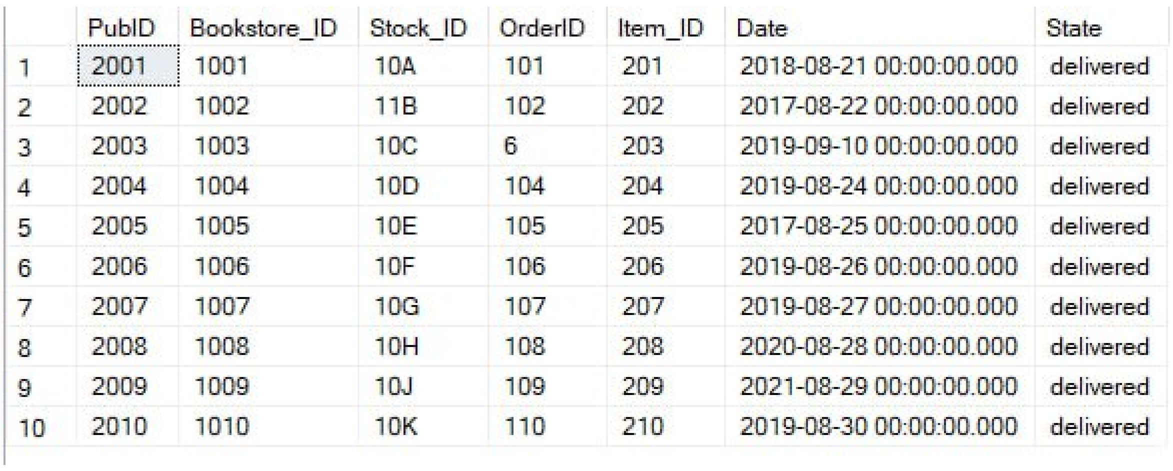
	OrderID	Order_Date_Time	Shipping_address	CID
1	-100	2018-08-01 00:00:00.000	Hall 11	3001
2	0	2022-08-01 00:00:00.000	Hall 11	3001
3	1	2018-08-02 00:00:00.000	Hall 12	3002
4	2	2018-09-03 00:00:00.000	Hall 13	3003
5	3	2018-06-04 00:00:00.000	Hall 14	3004
6	4	2018-05-05 00:00:00.000	Hall 15	3005
7	5	2018-04-06 00:00:00.000	Hall 16	3006
8	6	2018-04-07 00:00:00.000	Hall 17	3007
9	7	2018-03-08 00:00:00.000	Hall 11	3001
10	8	2018-02-09 00:00:00.000	Hall 12	3002
11	9	2018-01-10 00:00:00.000	Hall 13	3003
12	10	2018-06-11 00:00:00.000	Hall 14	3004
13	11	2018-07-12 00:00:00.000	Hall 15	3005
14	12	2018-09-12 00:00:00.000	Hall 16	3006
15	13	2018-09-13 00:00:00.000	Hall 17	3007
16	101	2022-08-02 00:00:00.000	Hall 12	3002
17	102	2022-09-03 00:00:00.000	Hall 13	3003
18	103	2022-06-04 00:00:00.000	Hall 14	3004
19	104	2022-05-05 00:00:00.000	Hall 15	3005
20	105	2022-04-06 00:00:00.000	Hall 16	3006
21	106	2022-04-07 00:00:00.000	Hall 17	3007
22	107	2022-03-08 00:00:00.000	Hall 11	3001
23	108	2022-02-09 00:00:00.000	Hall 12	3002
24	109	2022-01-10 00:00:00.000	Hall 13	3003
25	110	2022-06-11 00:00:00.000	Hall 14	3004
26	111	2022-07-12 00:00:00.000	Hall 15	3005
27	112	2022-09-12 00:00:00.000	Hall 16	3006
28	113	2022-09-13 00:00:00.000	Hall 17	3007

ITEMS_IN_ORDERS



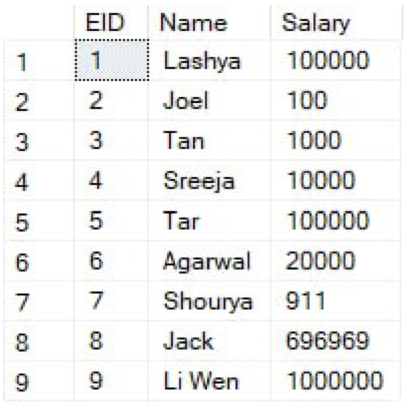
	PubID	Bookstore_ID	Stock_ID	OrderID	Item_ID	Item_Price	Item_Qty	Delivery_Date	CID	Comment	Rating	Feedback_Date_Time
1	2001	1001	10A	101	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
2	2001	1001	10A	102	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
3	2001	1001	10A	103	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
4	2001	1001	10A	104	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
5	2001	1001	10A	105	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
6	2001	1001	10A	106	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
7	2001	1001	10A	107	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
8	2001	1001	10A	108	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
9	2001	1001	10A	109	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
10	2001	1001	10A	110	201	35	1	2018-09-01 00:00:00.000	3001	good	5	2022-08-21 00:00:00.000
11	2002	1002	11B	102	202	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
12	2002	1002	11B	103	203	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
13	2002	1002	11B	104	204	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
14	2002	1002	11B	104	214	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
15	2002	1002	11B	105	205	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
16	2002	1002	11B	105	213	20	2	2017-09-02 00:00:00.000	3002	very good	4	2022-08-22 00:00:00.000
17	2002	1002	11B	106	206	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
18	2002	1002	11B	106	212	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
19	2002	1002	11B	107	207	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
20	2002	1002	11B	107	211	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
21	2002	1002	11B	108	208	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
22	2002	1002	11B	108	210	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
23	2002	1002	11B	109	209	20	2	2017-09-02 00:00:00.000	3002	very good	5	2022-08-22 00:00:00.000
24	2003	1003	10C	6	203	25.5	3	2018-09-03 00:00:00.000	3003	delivery ...	3	2018-09-23 00:00:00.000
25	2003	1004	10D	4	205	12	5	2019-09-10 00:00:00.000	3003	MEH	3	2019-09-10 00:00:00.000
26	2003	1005	10E	3	206	15	10	2019-09-10 00:00:00.000	3003	Not bad	2	2019-09-10 00:00:00.000
27	2004	1004	10D	104	204	33.9	4	2019-09-04 00:00:00.000	3004	good	4	2019-09-24 00:00:00.000
28	2005	1005	10E	105	205	28	5	2017-09-05 00:00:00.000	3005	bad qua...	2	2017-09-25 00:00:00.000
29	2006	1006	10F	106	206	26.9	6	2019-09-06 00:00:00.000	3006	really lik...	5	2022-08-26 00:00:00.000
30	2007	1007	10G	107	207	32.5	7	2019-09-07 00:00:00.000	3007	would n...	3	2019-09-27 00:00:00.000
31	2008	1008	10H	108	208	36	8	2020-09-08 00:00:00.000	3008	nice pro...	5	2020-09-28 00:00:00.000
32	2009	1009	10J	109	209	24	9	2021-09-09 00:00:00.000	3009	not so g...	3	2021-09-29 00:00:00.000
33	2010	1010	10K	110	210	22.5	10	2019-09-10 00:00:00.000	3010	good pri...	5	2019-09-30 00:00:00.000

ORDER_STATUS



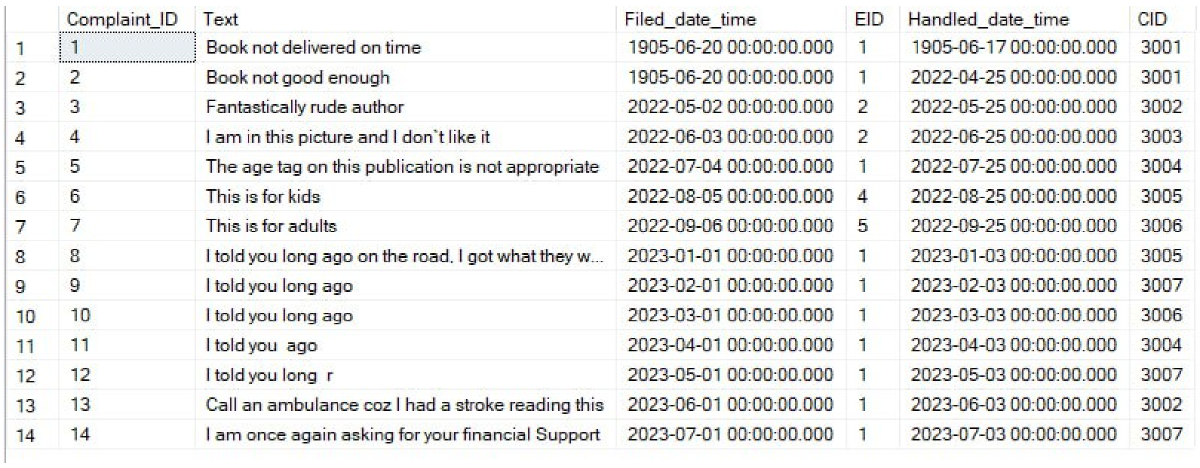
	PubID	Bookstore_ID	Stock_ID	OrderID	Item_ID	Date	State
1	2001	1001	10A	101	201	2018-08-21 00:00:00.000	delivered
2	2002	1002	11B	102	202	2017-08-22 00:00:00.000	delivered
3	2003	1003	10C	6	203	2019-09-10 00:00:00.000	delivered
4	2004	1004	10D	104	204	2019-08-24 00:00:00.000	delivered
5	2005	1005	10E	105	205	2017-08-25 00:00:00.000	delivered
6	2006	1006	10F	106	206	2019-08-26 00:00:00.000	delivered
7	2007	1007	10G	107	207	2019-08-27 00:00:00.000	delivered
8	2008	1008	10H	108	208	2020-08-28 00:00:00.000	delivered
9	2009	1009	10J	109	209	2021-08-29 00:00:00.000	delivered
10	2010	1010	10K	110	210	2019-08-30 00:00:00.000	delivered

EMPLOYEES



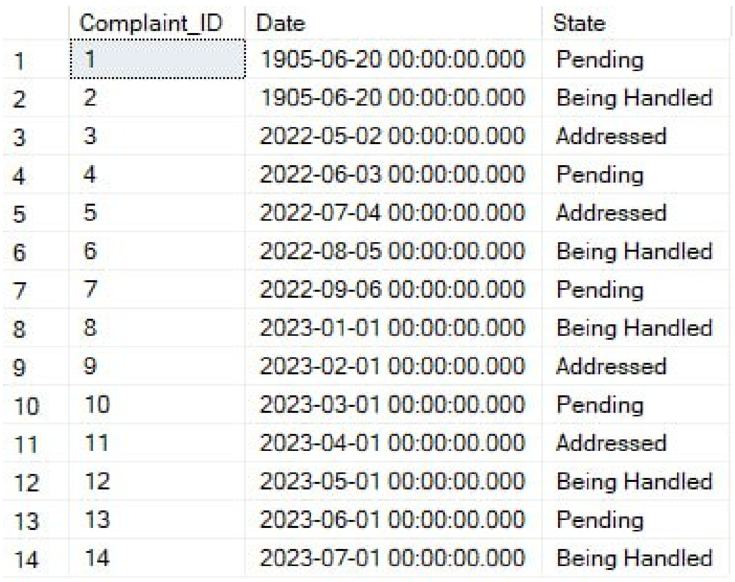
	EID	Name	Salary
1	1	Lashya	100000
2	2	Joel	100
3	3	Tan	1000
4	4	Sreeja	10000
5	5	Tar	100000
6	6	Agarwal	20000
7	7	Shourya	911
8	8	Jack	696969
9	9	Li Wen	1000000

COMPLAINTS



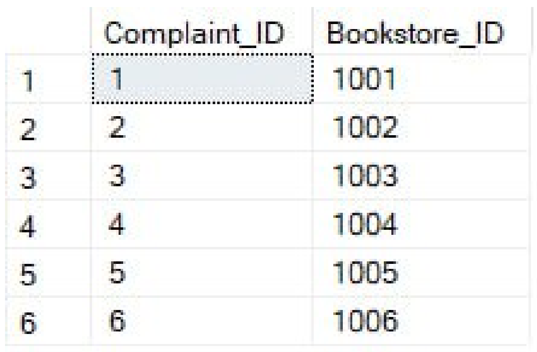
	Complaint_ID	Text	Filed_date_time	EID	Handled_date_time	CID
1	1	Book not delivered on time	1905-06-20 00:00:00.000	1	1905-06-17 00:00:00.000	3001
2	2	Book not good enough	1905-06-20 00:00:00.000	1	2022-04-25 00:00:00.000	3001
3	3	Fantastically rude author	2022-05-02 00:00:00.000	2	2022-05-25 00:00:00.000	3002
4	4	I am in this picture and I don`t like it	2022-06-03 00:00:00.000	2	2022-06-25 00:00:00.000	3003
5	5	The age tag on this publication is not appropriate	2022-07-04 00:00:00.000	1	2022-07-25 00:00:00.000	3004
6	6	This is for kids	2022-08-05 00:00:00.000	4	2022-08-25 00:00:00.000	3005
7	7	This is for adults	2022-09-06 00:00:00.000	5	2022-09-25 00:00:00.000	3006
8	8	I told you long ago on the road, I got what they w...	2023-01-01 00:00:00.000	1	2023-01-03 00:00:00.000	3005
9	9	I told you long ago	2023-02-01 00:00:00.000	1	2023-02-03 00:00:00.000	3007
10	10	I told you long ago	2023-03-01 00:00:00.000	1	2023-03-03 00:00:00.000	3006
11	11	I told you ago	2023-04-01 00:00:00.000	1	2023-04-03 00:00:00.000	3004
12	12	I told you long r	2023-05-01 00:00:00.000	1	2023-05-03 00:00:00.000	3007
13	13	Call an ambulance coz I had a stroke reading this	2023-06-01 00:00:00.000	1	2023-06-03 00:00:00.000	3002
14	14	I am once again asking for your financial Support	2023-07-01 00:00:00.000	1	2023-07-03 00:00:00.000	3007

COMPLAINT_STATUS



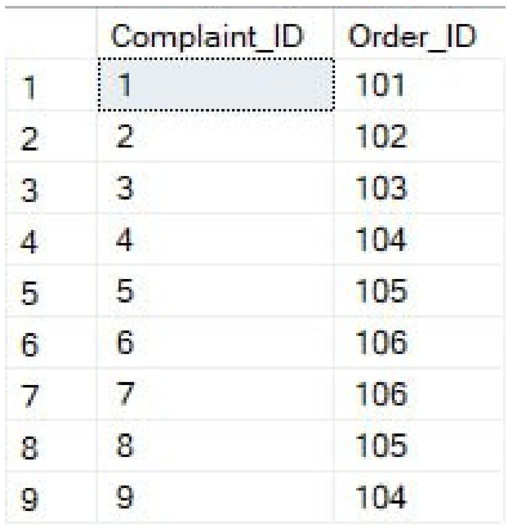
	Complaint_ID	Date	State
1	1	1905-06-20 00:00:00.000	Pending
2	2	1905-06-20 00:00:00.000	Being Handled
3	3	2022-05-02 00:00:00.000	Addressed
4	4	2022-06-03 00:00:00.000	Pending
5	5	2022-07-04 00:00:00.000	Addressed
6	6	2022-08-05 00:00:00.000	Being Handled
7	7	2022-09-06 00:00:00.000	Pending
8	8	2023-01-01 00:00:00.000	Being Handled
9	9	2023-02-01 00:00:00.000	Addressed
10	10	2023-03-01 00:00:00.000	Pending
11	11	2023-04-01 00:00:00.000	Addressed
12	12	2023-05-01 00:00:00.000	Being Handled
13	13	2023-06-01 00:00:00.000	Pending
14	14	2023-07-01 00:00:00.000	Being Handled

COMPLAINTS_ON_BOOKSTORE



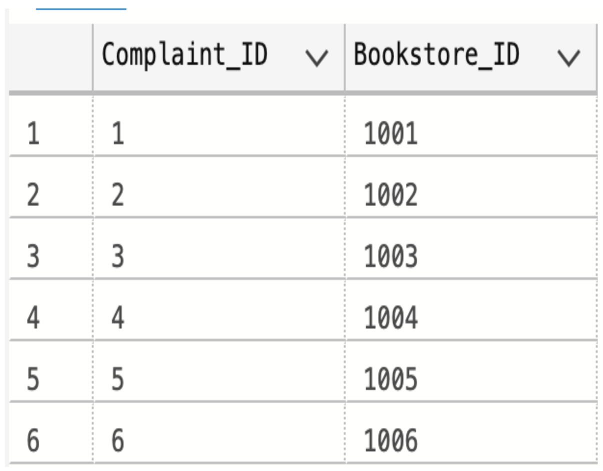
	Complaint_ID	Bookstore_ID
1	1	1001
2	2	1002
3	3	1003
4	4	1004
5	5	1005
6	6	1006

COMPLAINTS_ON_ORDER



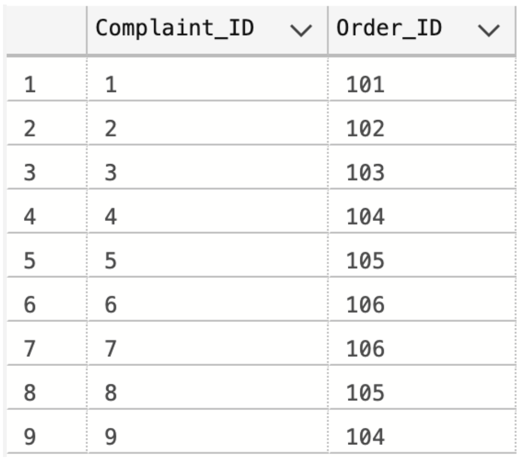
	Complaint_ID	Order_ID
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	106
7	7	106
8	8	105
9	9	104

COMPLAINTS_ON_BOOKSTORE:



	Complaint_ID	Bookstore_ID
1	1	1001
2	2	1002
3	3	1003
4	4	1004
5	5	1005
6	6	1006

COMPLAINTS_ON_ORDER:



	Complaint_ID	Order_ID
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	106
7	7	106
8	8	105
9	9	104

APPENDIX C: INDIVIDUAL CONTRIBUTION FORM

Full Name	**Individual Contribution to Lab 3 Submission**	**Percentage of Contribution**	**Signature**
Joel Tan	● Checked and edited the create commands for the creation of tables ● Checked and executed the population of tables, ensuring meaningful output for the queries ● Generated and tested the queries ● Recorded the demo showing the execution of the queries	30.00%	[illegible]
Tar Sreeja	● Checked the creation of tables ● Checked the population of tables ● Generated and checked the queries ● Checked the query outputs	25.00%	Sreeja Tar
Liu Liwen	● Created tables and populated the tables on price history, items in order and order	18%	Liu Liwen

	status ● Check the creation of tables ● Checked queries		
Agarwal Lakshya	● Created and populated some tables ● Checked and corrected some Queries ● Changed data according to queries	20%	Lakshya
Luo Minjuan	● Created tables and populated the tables on Publication, Bookstore, Customers, Stocks-In-Bookstore, Books, Magazines ● Check the creation of tables	16.67%	Luo Minjuan
Shourya Kuchhal	● Created tables ● Populated tables ● Changed data according to queries.	18%	Shourya Kuchhal