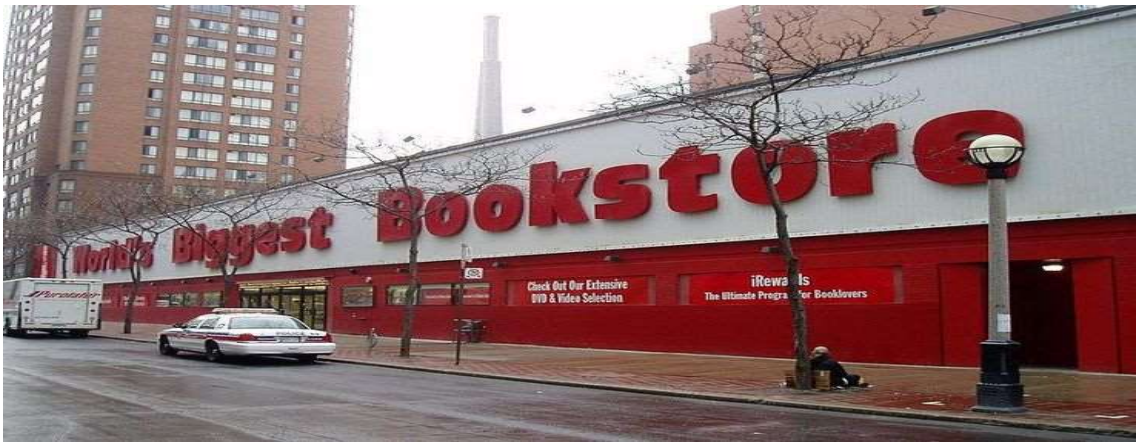
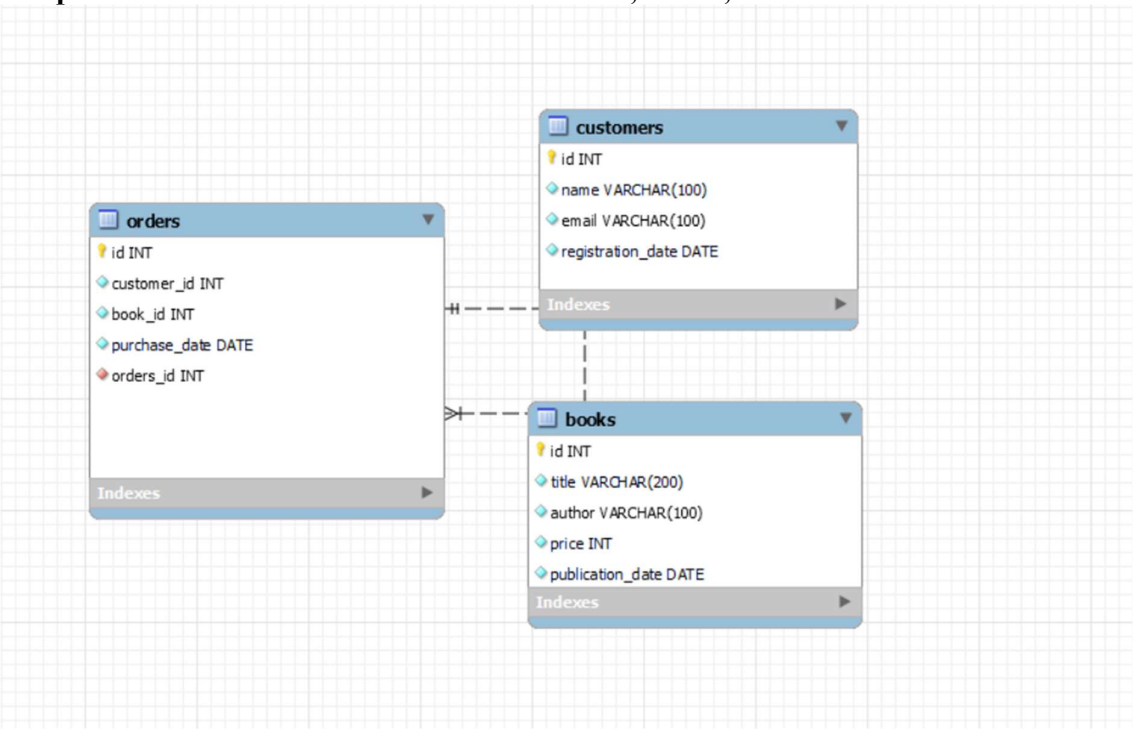
Analysis of Online Book Store Record and Retrieving the Max. Profit using SQL



We have the following online bookstore record in the form of table :
Customers: This table contains the customer's ID, name, email address, and the date they registered.
Books: This table contains the book's ID, title, author, price, and the date it was published.
Orders: This table contains the order's ID, customer ID, book ID, and the date it was purchased.

the queries to create the tables for the Customers, Books, and Orders tables:

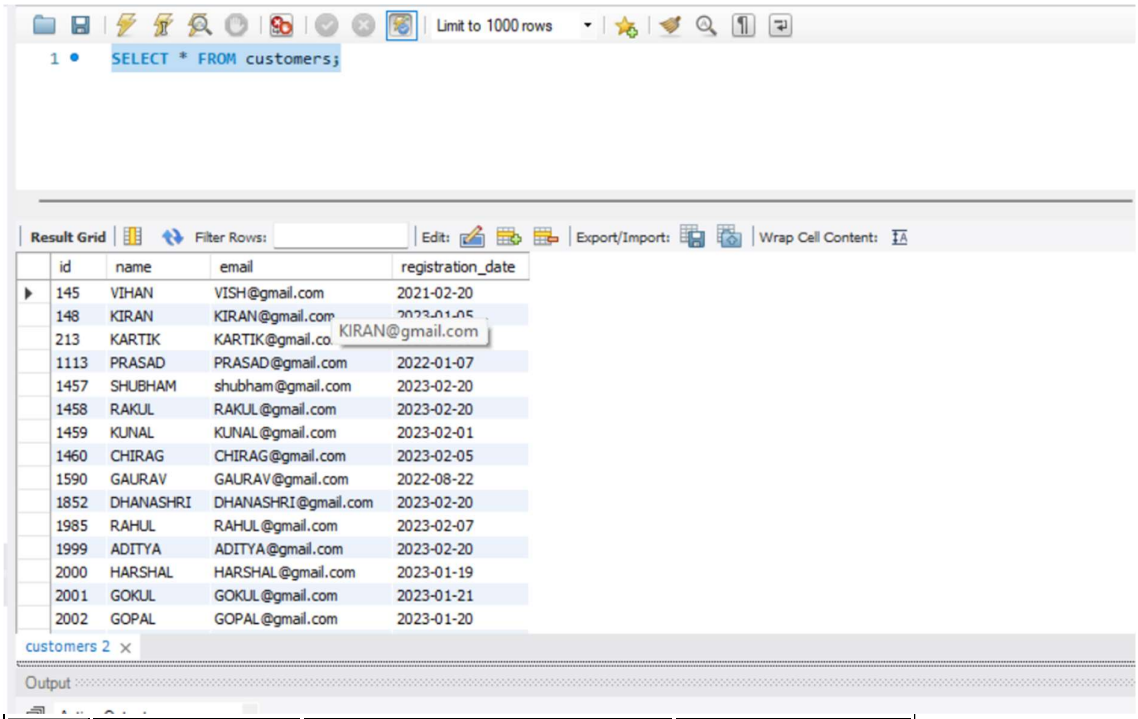


```
CREATE TABLE Customers (
  id int PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  registration_date DATE NOT NULL
);

CREATE TABLE Books (
  id INT PRIMARY KEY,
  title VARCHAR(200) NOT NULL,
  author VARCHAR(100) NOT NULL,
  price INT NOT NULL,
  publication_date DATE NOT NULL
);

CREATE TABLE Orders (
  id INT PRIMARY KEY,
  customer_id INT NOT NULL REFERENCES Customers(id),
  book_id INT NOT NULL REFERENCES Books(id),
  purchase_date DATE NOT NULL
);

INSERT INTO customers
(ID, name, email,registration_date)
values
(1999,"ADITYA","ADITYA@gmail.com","2023-02-20"),
(1458,"RAKUL","RAKUL@gmail.com","2023-02-20"),
(1459,"KUNAL","KUNAL@gmail.com","2023-02-01"),
(1460,"CHIRAG","CHIRAG@gmail.com","2023-02-05"),
(1985,"RAHUL","RAHUL@gmail.com","2023-02-07"),
(7896,"RUTUJA","RUTUJA@gmail.com","2023-02-08"),
(4563,"GAURI","GAURI@gmail.com","2023-02-10"),
(8520,"PRATIBHA","PRATIBHA@gmail.com","2023-01-22"),
(7412,"UDAY","UDAY@gmail.com","2023-02-01"),
(7530,"VISHAL","VISHAL@gmail.com","2022-02-05"),
(1590,"GAURAV","GAURAV@gmail.com","2022-08-22"),
(7562,"GAURANG","GAURANG@gmail.com","2021-01-23"),
(4561,"PRANAV","PRANAV@gmail.com","2022-02-20"),
(1852,"DHANASHRI","DHANASHRI@gmail.com","2023-02-20"),
(0145,"VISH","VISH@gmail.com","2021-02-20"),
(7584,"RUPALI","RUPALI@gmail.com","2020-02-20"),
(4896,"ROHIT","ROHIT@gmail.com","2021-04-20"),
(2154,"RITURAJ","RITURAJ@gmail.com","2023-01-20"),
(4578,"APURVA","APURVA@gmail.com","2022-07-20"),
(7555,"RAM","RAM@gmail.com","2022-02-20"),
(4488,"RAGHAV","RAGHAV@gmail.com","2020-02-20"),
(7485,"SONALI","SONALI@gmail.com","2022-10-20"),
(8574,"AMRUTA","AMRUTA@gmail.com","2023-08-20"),
(8754,"VISHWAJEET","VISHWAJEET@gmail.com","2023-02-19"),
(9898,"RUSHIKESH","RUSHIKESH@gmail.com","2023-02-18"),
```

(2255,"NIKITA","NIKITA@gmail.com","2023-02-17"),
(2266,"SANVED","SANVED@gmail.com","2023-02-15"),
(9966,"RAJ","RAJ@gmail.com","2023-02-14"),
(6969,"VAISHNAVI","VAISHNAVI@gmail.com","2023-02-12"),
(6565,"ABHIJEET","ABHIJEET@gmail.com","2023-02-11"),
(6363,"ABHIMANUE","ABHIMANUE@gmail.com","2023-02-10"),
(3636,"AISHWARYA","AISHWARYA@gmail.com","2023-02-09"),
(3583,"AKANSHA","AKANSHA@gmail.com","2023-02-08"),
(3654,"AKASH","AKASH@gmail.com","2023-02-07"),
(3658,"AKSHAY","AKSHAY@gmail.com","2023-02-06"),
(3689,"ANKITA","ANKITA@gmail.com","2023-02-08"),
(3674,"ANUSHKA","ANUSHKA@gmail.com","2023-02-20"),
(3677,"BALAJI","BALAJI@gmail.com","2023-02-05"),
(3611,"BHAVANA","BHAVANA@gmail.com","2023-02-04"),
(3600,"BHUSHAN","BHUSHAN@gmail.com","2023-02-03"),
(3612,"CHAITANYA","CHAITANYA@gmail.com","2023-02-02"),
(3622,"CHETAN","CHETAN@gmail.com","2023-01-30"),
(3659,"VAIBHAV","VAIBHAV@gmail.com","2023-01-28"),
(2500,"DEEP","DEEP@gmail.com","2023-01-28"),
(2501,"DEEPAK","DEEPAK@gmail.com","2023-01-27"),
(2502,"DHEERAJ","DHEERAJ@gmail.com","2023-02-26"),
(5036,"DIKSHANT","DIKSHANT@gmail.com","2023-01-25"),
(2504,"DIPALI","DIPALI@gmail.com","2023-01-24"),
(2505,"GANESH","GANESH@gmail.com","2023-01-23"),
(2506,"GEETANJALI","GEETANJALI@gmail.com","2023-01-22"),
(2001,"GOKUL","GOKUL@gmail.com","2023-01-21"),
(2002,"GOPAL","GOPAL@gmail.com","2023-01-20"),
(2000,"HARSHAL","HARSHAL@gmail.com","2023-01-19"),
(2003,"HEMANT","HEMANT@gmail.com","2023-01-18"),
(2004,"ISHA","ISHA@gmail.com","2023-01-17"),
(2005,"JAGDISH","JAGDISH@gmail.com","2022-01-16"),
(8842,"JAVERIYA","JAVERIYA@gmail.com","2023-01-15"),
(8843,"JAYA","JAYA@gmail.com","2023-01-14"),
(8844,"JEYESH","JEYESH@gmail.com","2023-01-12"),
(8846,"KAJAL","KAJAL@gmail.com","2023-01-11"),
(88874,"KANCHAN","KANCHAN@gmail.com","2023-01-10"),
(0213,"KARTIK","KARTIK@gmail.com","2023-01-09"),
(8968,"KEDAR","KEDAR@gmail.com","2023-02-08"),
(0148,"KIRAN","KIRAN@gmail.com","2023-01-05"),
(7474,"KISHOR","KISHOR@gmail.com","2023-01-04"),
(7410,"KOMAL","KOMAL@gmail.com","2023-02-03"),
(7445,"LAKHAN","LAKHAN@gmail.com","2023-01-02"),
(7411,"LAVANYA","LAVANYA@gmail.com","2023-01-01"),
(7416,"MADHU","MADHU@gmail.com","2022-10-24"),
(7419,"MOHINI","MOHINI@gmail.com","2022-10-26"),
(7488,"NIKHIL","NIKHIL@gmail.com","2022-11-24"),
(3597,"NAINA","NAINA@gmail.com","2023-11-21"),
(3976,"NILESH","NILESH@gmail.com","202-11-20"),
(7569,"NILKANTH","NILKANTH@gmail.com","2023-02-20"),
(9998,"PANKAJ","PANKAJ@gmail.com","2023-02-20"),

(9999,"PARESH","PARESH@gmail.com","2023-02-20"),
(88883,"PRANALI","PRANALI@gmail.com","2023-02-20"),
(1113,"PRASAD","PRASAD@gmail.com","2022-01-07");

 # 1. List all the customers and their details.
SELECT id, name, email, registration_date FROM customers;
select * from customers;



id	name	Email	Registration_date
145	VIHAN	VISH@gmail.com	2021-02-20
148	KIRAN	KIRAN@gmail.com	2023-01-05
213	KARTIK	KARTIK@gmail.com	2023-01-09
1113	PRASAD	PRASAD@gmail.com	2022-01-07
1457	SHUBHAM	shubham@gmail.com	2023-02-20
1458	RAKUL	RAKUL@gmail.com	2023-02-20
1459	KUNAL	KUNAL@gmail.com	2023-02-01
1460	CHIRAG	CHIRAG@gmail.com	2023-02-05
1590	GAURAV	GAURAV@gmail.com	2022-08-22
1852	DHANASHRI	DHANASHRI@gmail.com	2023-02-20
1985	RAHUL	RAHUL@gmail.com	2023-02-07
1999	ADITYA	ADITYA@gmail.com	2023-02-20
2000	HARSHAL	HARSHAL@gmail.com	2023-01-19
2001	GOKUL	GOKUL@gmail.com	2023-01-21
2002	GOPAL	GOPAL@gmail.com	2023-01-20

2003	HEMANT	HEMANT@gmail.com	2023-01-18
2004	ISHA	ISHA@gmail.com	2023-01-17
2005	JAGDISH	JAGDISH@gmail.com	2022-01-16
2154	RITURAJ	RITURAJ@gmail.com	2023-01-20
2255	NIKITA	NIKITA@gmail.com	2023-02-17
2266	SANVED	SANVED@gmail.com	2023-02-15
2500	DEEP	DEEP@gmail.com	2023-01-28
2501	DEEPAK	DEEPAK@gmail.com	2023-01-27
2502	DHEERAJ	DHEERAJ@gmail.com	2023-02-26
2504	DIPALI	DIPALI@gmail.com	2023-01-24
2505	GANESH	GANESH@gmail.com	2023-01-23
2506	GEETANJALI	GEETANJALI@gmail.com	2023-01-22
3583	AKANSHA	AKANSHA@gmail.com	2023-02-08
3597	NAINA	NAINA@gmail.com	2023-11-21
3600	BHUSHAN	BHUSHAN@gmail.com	2023-02-03
3611	BHAVANA	BHAVANA@gmail.com	2023-02-04
3612	CHAITANYA	CHAITANYA@gmail.com	2023-02-02
3622	CHETAN	CHETAN@gmail.com	2023-01-30
3636	AISHWARYA	AISHWARYA@gmail.com	2023-02-09
3654	AKASH	AKASH@gmail.com	2023-02-07
3658	AKSHAY	AKSHAY@gmail.com	2023-02-06
3659	VAIBHAV	VAIBHAV@gmail.com	2023-01-28
3674	ANUSHKA	ANUSHKA@gmail.com	2023-02-20
3677	BALAJI	BALAJI@gmail.com	2023-02-05
3689	ANKITA	ANKITA@gmail.com	2023-02-08
3976	NILESH	NILESH@gmail.com	0202-11-20
4488	RAGHAV	RAGHAV@gmail.com	2020-02-20
4561	PRANAV	PRANAV@gmail.com	2022-02-20
4563	GAURI	GAURI@gmail.com	2023-02-10
4578	APURVA	APURVA@gmail.com	2022-07-20
4896	ROHIT	ROHIT@gmail.com	2021-04-20
5036	DIKSHANT	DIKSHANT@gmail.com	2023-01-25
6363	ABHIMANUE	ABHIMANUE@gmail.com	2023-02-10
6565	ABHIJEET	ABHIJEET@gmail.com	2023-02-11
6969	VAISHNAVI	VAISHNAVI@gmail.com	2023-02-12
7410	KOMAL	KOMAL@gmail.com	2023-02-03
7411	LAVANYA	LAVANYA@gmail.com	2023-01-01
7412	UDAY	UDAY@gmail.com	2023-02-01
7416	MADHU	MADHU@gmail.com	2022-10-24

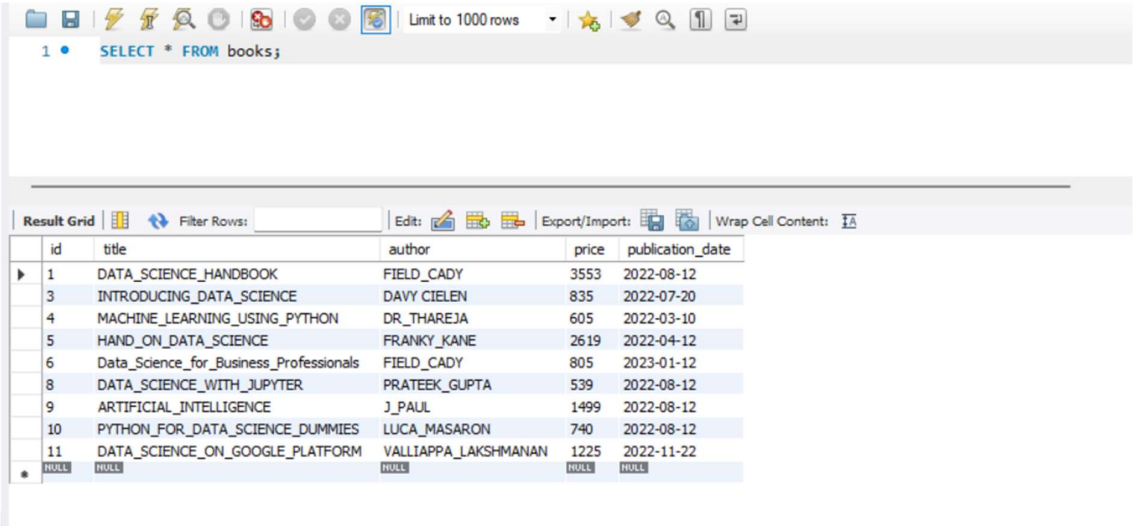
7419	MOHINI	MOHINI@gmail.com	2022-10-26
7445	LAKHAN	LAKHAN@gmail.com	2023-01-02
7474	KISHOR	KISHOR@gmail.com	2023-01-04
7485	SONALI	SONALI@gmail.com	2022-10-20
7488	NIKHIL	NIKHIL@gmail.com	2022-11-24
7530	VISHAL	VISHAL@gmail.com	2022-02-05
7555	RAM	RAM@gmail.com	2022-02-20
7562	GAURANG	GAURANG@gmail.com	2021-01-23
7569	NILKANTH	NILKANTH@gmail.com	2023-02-20
7584	RUPALI	RUPALI@gmail.com	2020-02-20
7896	RUTUJA	RUTUJA@gmail.com	2023-02-08
8520	PRATIBHA	PRATIBHA@gmail.com	2023-01-22
8574	AMRUTA	AMRUTA@gmail.com	2023-08-20
8754	VISHWAJEET	VISHWAJEET@gmail.com	2023-02-19
8842	JAVERIYA	JAVERIYA@gmail.com	2023-01-15
8843	JAYA	JAYA@gmail.com	2023-01-14
8844	JEYESH	JEYESH@gmail.com	2023-01-12
8846	KAJAL	KAJAL@gmail.com	2023-01-11
8968	KEDAR	KEDAR@gmail.com	2023-02-08
9137	PRAKASH	PRAKASH@GMAIL.COM	2020-11-30
9785	abhay	abhay@gmail.com	2019-08-25
9898	RUSHIKESH	RUSHIKESH@gmail.com	2023-02-18
9966	RAJ	RAJ@gmail.com	2023-02-14
9998	PANKAJ	PANKAJ@gmail.com	2023-02-20
88874	KANCHAN	KANCHAN@gmail.com	2023-01-10
88883	PRANALI	PRANALI@gmail.com	2023-02-20

```
INSERT INTO BOOKS(
ID, title, author, price,publication_date)
values
(0001,"DATA_SCIENCE_HANDBOOK","FIELD_CADY",3553,"2022-08-12"),
(0003,"INTRODUCING_DATA_SCIENCE","DAVY CIELEN",835,"2022-07-20"),
(0004,"MACHINE_LEARNING_USING_PYTHON","DR_THAREJA",605,"2022-03-10"),
(0005,"HAND_ON_DATA_SCIENCE","FRANKY_KANE",2619,"2022-04-12"),
(0006,"Data_Science_for_Business_Professionals","FIELD_CADY",805,"2023-01-12"),
(0008,"DATA_SCIENCE_WITH_JUPYTER","PRATEEK_GUPTA",539,"2022-08-12"),
(0009,"ARTIFICIAL_INTELLIGENCE","J_PAUL",1499,"2022-08-12"),
(0010,"PYTHON_FOR_DATA_SCIENCE_DUMMIES","LUCA_MASARON",740,"2022-08-12"),
(0011,"DATA_SCIENCE_ON_GOOGLE_PLATFORM","VALLIAPPA_LAKSHMANAN",1225,"2022-11-22");
```

 # 2. List all the books in the database and their authors.

SELECT title, author FROM books;

select * from books;



```
SELECT * FROM books;
```

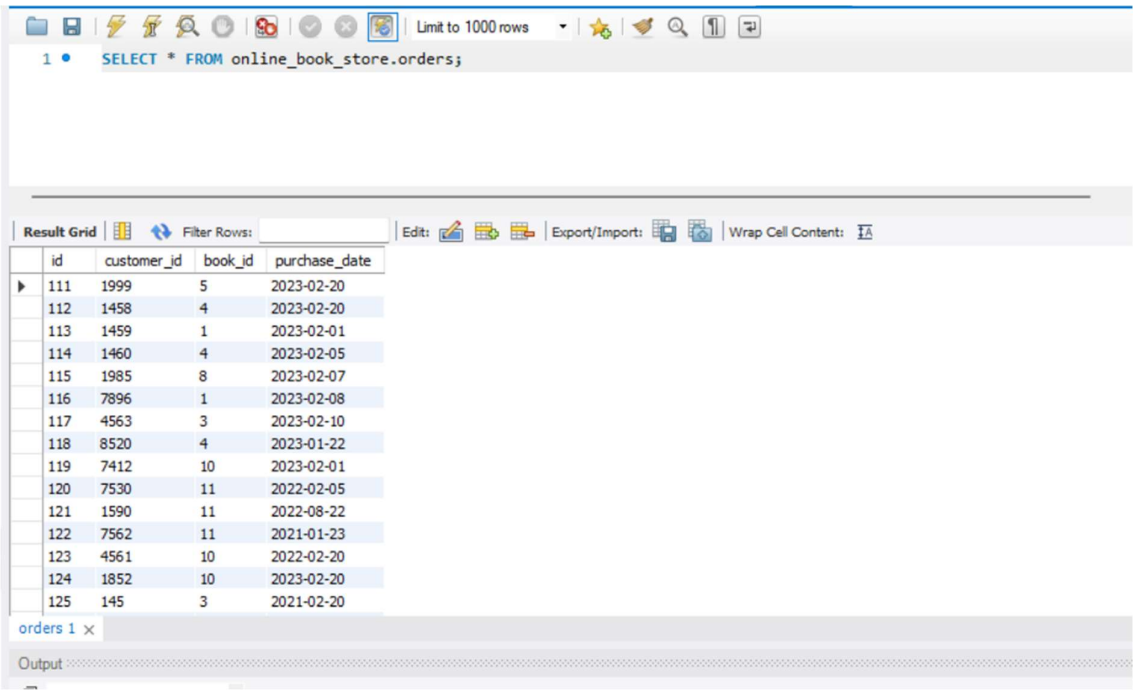
id	title	author	price	publication_date
1	DATA_SCIENCE_HANDBOOK	FIELD_CADY	3553	2022-08-12
3	INTRODUCING_DATA_SCIENCE	DAVY CIELEN	835	2022-07-20
4	MACHINE_LEARNING_USING_PYTHON	DR_THAREJA	605	2022-03-10
5	HAND_ON_DATA_SCIENCE	FRANKY_KANE	2619	2022-04-12
6	Data_Science_for_Business_Professionals	FIELD_CADY	805	2023-01-12
8	DATA_SCIENCE_WITH_JUPYTER	PRATEEK_GUPTA	539	2022-08-12
9	ARTIFICIAL_INTELLIGENCE	J_PAUL	1499	2022-08-12
10	PYTHON_FOR_DATA_SCIENCE_DUMMIES	LUCA_MASARON	740	2022-08-12
11	DATA_SCIENCE_ON_GOOGLE_PLATFORM	VALLIAPPA_LAKSHMANAN	1225	2022-11-22
NULL	NULL	NULL	NULL	NULL

INSERT INTO ORDERS(
ID, customer_ID, book_ID,purchase_date)
values
(111,1999,005,"2023-02-20"),
(112,1458,004,"2023-02-20"),
(113,1459,001,"2023-02-01"),
(114,1460,004,"2023-02-05"),
(115,1985,008,"2023-02-07"),
(116,7896,001,"2023-02-08"),
(117,4563,003,"2023-02-10"),
(118,8520,004,"2023-01-22"),
(119,7412,010,"2023-02-01"),
(120,7530,011,"2022-02-05"),
(121,1590,011,"2022-08-22"),
(122,7562,011,"2021-01-23"),
(123,4561,010,"2022-02-20"),
(124,1852,010,"2023-02-20"),
(125,0145,003,"2021-02-20"),
(126,7584,003,"2020-02-20"),
(127,4896,003,"2021-04-20"),
(128,2154,003,"2023-01-20"),
(129,4578,009,"2022-07-20"),
(130,7555,009,"2022-02-20"),
(131,4488,009,"2020-02-20"),
(132,7485,009,"2022-10-20"),
(133,8574,001,"2023-08-20"),

(134,8754,001,"2023-02-19"),
(135,9898,001,"2023-02-18"),
(136,2255,003,"2023-02-17"),
(137,2266,001,"2023-02-15"),
(138,9966,004,"2023-02-14"),
(139,6969,004,"2023-02-12"),
(140,6565,004,"2023-02-11"),
(141,6363,001,"2023-02-10"),
(142,3636,001,"2023-02-09"),
(143,3583,001,"2023-02-08"),
(144,3654,003,"2023-02-07"),
(145,3658,003,"2023-02-06"),
(146,3689,003,"2023-02-08"),
(147,3674,003,"2023-02-20"),
(148,3677,003,"2023-02-05"),
(149,3611,004,"2023-02-04"),
(150,3600,004,"2023-02-03"),
(151,3612,005,"2023-02-02"),
(152,3622,005,"2023-01-30"),
(153,3659,005,"2023-01-28"),
(155,2500,005,"2023-01-28"),
(156,2501,005,"2023-01-27"),
(157,2502,005,"2023-02-26"),
(158,5036,005,"2023-01-25"),
(159,2504,009,"2023-01-24"),
(160,2505,009,"2023-01-23"),
(161,2506,009,"2023-01-22"),
(162,2001,009,"2023-01-21"),
(163,2002,009,"2023-01-20"),
(164,2000,001,"2023-01-19"),
(165,2003,001,"2023-01-18"),
(166,2004,001,"2023-01-17"),
(167,2005,003,"2022-01-16"),
(168,8842,001,"2023-01-15"),
(169,8843,003,"2023-01-14"),
(170,8844,004,"2023-01-12"),
(171,8846,004,"2023-01-11"),
(172,88874,004,"2023-01-10"),
(173,0213,005,"2023-01-09"),
(174,8968,005,"2023-02-08"),
(175,0148,005,"2023-01-05"),
(176,474,005,"2023-01-04"),
(177,7410,006,"2023-02-03"),
(178,7445,006,"2023-01-02"),
(179,7411,006,"2023-01-01"),
(180,7416,008,"2022-10-24"),
(181,7419,006,"2022-10-26"),
(182,7488,008,"2022-11-24"),
(183,3597,008,"2023-11-21"),
(184,3976,010,"202-11-20"),

(185,7569,011,"2023-02-20"),
(186,9998,011,"2023-02-20"),
(187,9999,011,"2023-02-20"),
(188,88883,011,"2023-02-20"),
(189,1113,010,"2022-01-07");

 # 3. List all the orders and their details.
SELECT * FROM orders;



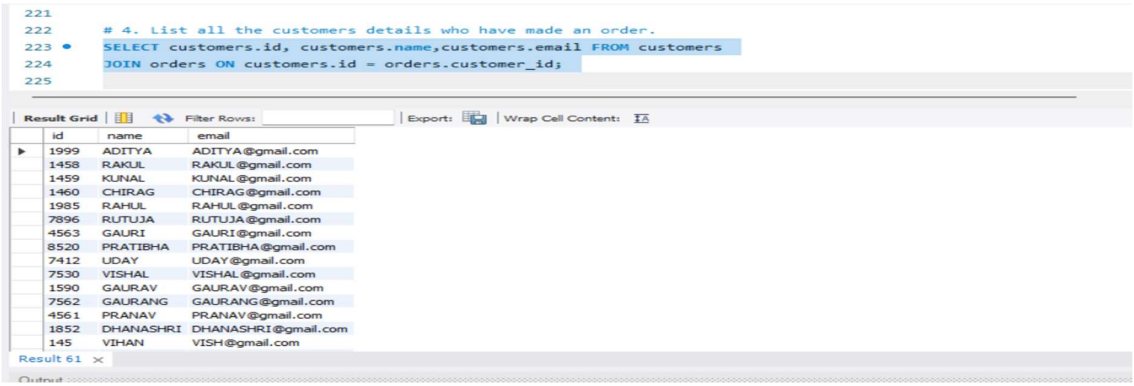
id	customer_id	book_id	purchase_date
111	1999	5	2023-02-20
112	1458	4	2023-02-20
113	1459	1	2023-02-01
114	1460	4	2023-02-05
115	1985	8	2023-02-07
116	7896	1	2023-02-08
117	4563	3	2023-02-10
118	8520	4	2023-01-22
119	7412	10	2023-02-01
120	7530	11	2022-02-05
121	1590	11	2022-08-22
122	7562	11	2021-01-23
123	4561	10	2022-02-20
124	1852	10	2023-02-20
125	145	3	2021-02-20

124	1852	10	2023-02-20
125	145	3	2021-02-20
126	7584	3	2020-02-20
127	4896	3	2021-04-20
128	2154	3	2023-01-20
129	4578	9	2022-07-20
130	7555	9	2022-02-20
131	4488	9	2020-02-20
132	7485	9	2022-10-20
133	8574	1	2023-08-20
134	8754	1	2023-02-19
135	9898	1	2023-02-18
136	2255	3	2023-02-17
137	2266	1	2023-02-15
138	9966	4	2023-02-14
139	6969	4	2023-02-12
140	6565	4	2023-02-11
141	6363	1	2023-02-10
142	3636	1	2023-02-09
143	3583	1	2023-02-08
144	3654	3	2023-02-07
145	3658	3	2023-02-06
146	3689	3	2023-02-08
147	3674	3	2023-02-20
148	3677	3	2023-02-05
149	3611	4	2023-02-04
150	3600	4	2023-02-03
151	3612	5	2023-02-02
152	3622	5	2023-01-30
153	3659	5	2023-01-28
155	2500	5	2023-01-28
156	2501	5	2023-01-27
157	2502	5	2023-02-26
158	5036	5	2023-01-25
159	2504	9	2023-01-24
160	2505	9	2023-01-23
161	2506	9	2023-01-22
162	2001	9	2023-01-21
163	2002	9	2023-01-20

164	2000	1	2023-01-19
165	2003	1	2023-01-18
166	2004	1	2023-01-17
167	2005	3	2022-01-16
168	8842	1	2023-01-15
169	8843	3	2023-01-14
170	8844	4	2023-01-12
171	8846	4	2023-01-11
172	88874	4	2023-01-10
173	213	5	2023-01-09
174	8968	5	2023-02-08
175	148	5	2023-01-05
176	474	5	2023-01-04
177	7410	6	2023-02-03
178	7445	6	2023-01-02
179	7411	6	2023-01-01
180	7416	8	2022-10-24
181	7419	6	2022-10-26
182	7488	8	2022-11-24
183	3597	8	2023-11-21
184	3976	10	0202-11-20
185	7569	11	2023-02-20
186	9998	11	2023-02-20
187	9999	11	2023-02-20
188	88883	11	2023-02-20
189	1113	10	2022-01-07

 # 4. List all the customers who have made an order.

SELECT customers.id, customers.name FROM customers
JOIN orders ON customers.id = orders.customer_id;



```
# 4. List all the customers details who have made an order.
SELECT customers.id, customers.name,customers.email FROM customers
JOIN orders ON customers.id = orders.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	email
1999	ADITYA	ADITYA@gmail.com
1458	RAKUL	RAKUL@gmail.com
1459	KUNAL	KUNAL@gmail.com
1460	CHIRAG	CHIRAG@gmail.com
1985	RAHUL	RAHUL@gmail.com
7896	RUTUJA	RUTUJA@gmail.com
4563	GAURI	GAURI@gmail.com
8520	PRATIBHA	PRATIBHA@gmail.com
7412	UDAY	UDAY@gmail.com
7530	VISHAL	VISHAL@gmail.com
1590	GAURAV	GAURAV@gmail.com
7562	GAURANG	GAURANG@gmail.com
4561	PRANAV	PRANAV@gmail.com
1852	DHANASHRI	DHANASHRI@gmail.com
145	VIHAN	VISH@gmail.com

Result 61

-- after inserting all data we found that there was one csutomer who visted bookstore but not on record.

 # 5. add the details of customer whose name is "abhay".

insert into customers
(ID, name, email,registration_date)
values
(9085,"ayush","ayush@gmail.com","2019-08-03");
Select * from customers;

```
# 5. add the details of customer whose name is "abhay".
insert into customers
(ID, name, email,registration_date)
values
(9085,"ayush","yush@gmail.com","2019-08-03");
select * from customers;

# 6. update the name of customer as "     " whose name where mentioned as "     " where id is "     ".
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date
8754	VISHWAJEET	VISHWAJEET@gmail.com	2023-02-19
8842	JAVERIYA	JAVERIYA@gmail.com	2023-01-15
8843	JAYA	JAYA@gmail.com	2023-01-14
8844	JEYESH	JEYESH@gmail.com	2023-01-12
8846	KAJAL	KAJAL@gmail.com	2023-01-11
8968	KEDAR	KEDAR@gmail.com	2023-02-08
9085	ayush	yush@gmail.com	2019-08-03
9137	PRAKASH	PRAKASH@GMAIL.COM	2020-11-30
9785	abhay	abhay@gmail.com	2019-08-25
9898	RUSHIKESH	RUSHIKESH@gmail.com	2023-02-18
9966	RAJ	RAJ@gmail.com	2023-02-14

customers 62

 # 6. update the name of customer as " VARUN "where id is " 145".

UPDATE CUSTOMERS
SET name = "VARUN"
WHERE id = 145;
select * from customers;

```
# 6. update the name of customer as "VARUN" whose id is "145".
UPDATE CUSTOMERS
SET name = "VARUN"
WHERE id = 145;
select * from customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date
145	VARUN	VISH@gmail.com	2021-02-20
148	KIRAN	KIRAN@gmail.com	2023-01-05
213	KARTIK	KARTIK@gmail.com	2023-01-09
1113	PRASAD	PRASAD@gmail.com	2022-01-07
1457	SHUBHAM	shubham@gmail.com	2023-02-20
1458	RAKUL	RAKUL@gmail.com	2023-02-20
1459	KUNAL	KUNAL@gmail.com	2023-02-01
1460	CHIRAG	CHIRAG@gmail.com	2023-02-05
1590	GAURAV	GAURAV@gmail.com	2022-08-22
1852	DHANASHRI	DHANASHRI@gmail.com	2023-02-20
1985	RAHUL	RAHUL@gmail.com	2023-02-07

-- after analyasing table we found that customer name "paresh" had not visited bookstore.

 # 7. delete record of customer having their name as paresh.

delete from customers
where name = "paresh";
select * from customers;

```
-- after analyasing table we found that customer name "paresh" had not visited bookstore.
# 7. delete record of customer having their name as paresh.
delete from customers
where name = "paresh";
select * from customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date
145	VARUN	VISH@gmail.com	2021-02-20
148	KIRAN	KIRAN@gmail.com	2023-01-05
213	KARTIK	KARTIK@gmail.com	2023-01-09
1113	PRASAD	PRASAD@gmail.com	2022-01-07
1457	SHUBHAM	shubham@gmail.com	2023-02-20
1458	RAKUL	RAKUL@gmail.com	2023-02-20
1459	KUNAL	KUNAL@gmail.com	2023-02-01
1460	CHIRAG	CHIRAG@gmail.com	2023-02-05
1590	GAURAV	GAURAV@gmail.com	2022-08-22
1852	DHANASHRI	DHANASHRI@gmail.com	2023-02-20
1985	RAHUL	RAHUL@gmail.com	2023-02-07

customers 63

-- after retriving all data, we found that there should be a column name "gender" in customer.

 # 8. add a new column name "gender" in customers table:

alter table customers
add column gender varchar(10);
select * from customers;

```
-- after retriving all data, we found that there should be a column name "gender" in customer.
# 8. add a new column name "gender" in customers table:
alter table customers
add column gender varchar(10);
select * from customers;

# 9. delete column gender as we have no data available.
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date	gender
145	VARUN	VISH@gmail.com	2021-02-20	NULL
148	KIRAN	KIRAN@gmail.com	2023-01-05	NULL
213	KARTIK	KARTIK@gmail.com	2023-01-09	NULL
1113	PRASAD	PRASAD@gmail.com	2022-01-07	NULL
1457	SHUBHAM	shubham@gmail.com	2023-02-20	NULL
1458	RAKUL	RAKUL@gmail.com	2023-02-20	NULL
1459	KUNAL	KUNAL@gmail.com	2023-02-01	NULL
1460	CHIRAG	CHIRAG@gmail.com	2023-02-05	NULL
1590	GAURAV	GAURAV@gmail.com	2022-08-22	NULL
1852	DHANASHRI	DHANASHRI@gmail.com	2023-02-20	NULL
1985	RAHUL	RAHUL@gmail.com	2023-02-07	NULL

customers 64

9. delete column gender as we have no data available.

alter table customers
drop column gender;

```
# 9. delete column gender as we have no data available.
alter table customers
drop column gender;
SELECT * FROM CUSTOMERS;

# 10. find the book ordered by chirag and on which date
SELECT books.title,orders.id,customers.name,orders.purchase_date FROM orders
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date
145	VARUN	VISH@gmail.com	2021-02-20
148	KIRAN	KIRAN@gmail.com	2023-01-05
213	KARTIK	KARTIK@gmail.com	2023-01-09
1113	PRASAD	PRASAD@gmail.com	2022-01-07
1457	SHUBHAM	shubham@gmail.com	2023-02-20
1458	RAKUL	RAKUL@gmail.com	2023-02-20
1459	KUNAL	KUNAL@gmail.com	2023-02-01
1460	CHIRAG	CHIRAG@gmail.com	2023-02-05
1590	GAURAV	GAURAV@gmail.com	2022-08-22
1852	DHANASHRI	DHANASHRI@gmail.com	2023-02-20
1985	RAHUL	RAHUL@gmail.com	2023-02-07

CUSTOMERS 65

Output

10. find the book ordered by chirag and on which date

SELECT books.title,orders.id,customers.name,orders.purchase_date FROM orders
JOIN customers ON orders.customer_id = customers.id
JOIN books ON orders.book_id = books.id
where customers.name = "chirag";

```
# 10. find the book ordered by chirag and on which date
SELECT books.title,orders.id,customers.name,orders.purchase_date FROM orders
JOIN customers ON orders.customer_id = customers.id
JOIN books ON orders.book_id = books.id
where customers.name = "chirag";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	id	name	purchase_date
MACHINE_LEARNING_USING_PYTHON	114	CHIRAG	2023-02-05

11. List all the orders along with the customer and book details.

SELECT orders.id, customers.name, books.title, orders.purchase_date FROM orders
JOIN customers ON orders.customer_id = customers.id
JOIN books ON orders.book_id = books.id;

```
# 11. List all the orders along with the customer and book details.
SELECT orders.id, customers.name, books.title, orders.purchase_date FROM orders
JOIN customers ON orders.customer_id = customers.id
JOIN books ON orders.book_id = books.id;

# 12. Retrieve all the books details whose price is below 1000
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	title	purchase_date
111	ADITYA	HAND_ON_DATA_SCIENCE	2023-02-20
112	RAKUL	MACHINE_LEARNING_USING_PYTHON	2023-02-20
113	KUNAL	DATA_SCIENCE_HANDBOOK	2023-02-01
114	CHIRAG	MACHINE_LEARNING_USING_PYTHON	2023-02-05
115	RAHUL	DATA_SCIENCE_WITH_JUPYTER	2023-02-07
116	RUTUJA	DATA_SCIENCE_HANDBOOK	2023-02-08
117	GAURI	INTRODUCING_DATA_SCIENCE	2023-02-10
118	PRATIBHA	MACHINE_LEARNING_USING_PYTHON	2023-01-22
119	UDAY	PYTHON_FOR_DATA_SCIENCE_DUMMIES	2023-02-01
120	VISHAL	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2022-02-05
121	GAURAV	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2022-08-22

Result 67

Output

12. Retrieve all the books details whose price is below 1000

select books.id,books.title,books.price from books
where price <1000;

```
# 12. Retrieve all the books details whose price is below 1000
select books.id,books.title,books.price from books
where price <1000;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	price
3	INTRODUCING_DATA_SCIENCE	835
4	MACHINE_LEARNING_USING_PYTHON	605
6	Data_Science_for_Business_Professionals	805
8	DATA_SCIENCE_WITH_JUPYTER	539
10	PYTHON_FOR_DATA_SCIENCE_DUMMIES	740
NULL	NULL	NULL

13. Retrieve the no. of books whose price is below 1000

select count(books.id) from books
where price <1000;

```
# 13. Retrieve the no. of books whose price is below 1000
select count(books.id) from books
where price <1000;

# 14. retrive the book which is ordered by rahul
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(books.id)
5

14. retrive the book which is ordered by rahul

select books.title,orders.id,customers.name from customers
join orders on orders.customer_id=customers.id
join books on orders.book_id=books.id
where name = "rahul";

```
# 6. retrive the book which is ordered by rahul
select books.title,orders.id,customers.name from customers
join orders on orders.customer_id=customers.id
join books on orders.book_id=books.id
where name = "rahul";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	id	name
DATA_SCIENCE_WITH_JUPYTER	115	RAHUL

15. retrieve the customers whose name start with their 2nd letter as "a".

select customers.name from customers
where name like "_a%" ;

```
# 18. retrieve the customers whose name start with their 2nd letter as "a".
select customers.name from customers
where name like "_a%" ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
VARUN
KARTIK
RAKUL
GAURAV
RAHUL
HARSHAL
JAGDISH
SANVED

customers 2

 # 16. retrieve the name of book whose name ending with "handbook".

select books.title from books where
title like "%handbook";

```
# 19. retrieve the name of book whose name ending with "handbook".
select books.title from books where
title like "%handbook";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title
DATA_SCIENCE_HANDBOOK

 # 17. retrieve tha name of book which is booked by rahul,vishal,rutuja,gauri;

select books.title,customers.name from customers
join orders on orders.customer_id=customers.id
join books on orders.book_id=books.id
where customers.name in("rahul","vishal","rutuja","gauri") ;

```
# 8. retrieve tha name of book which is booked by rahul,vishal,rutuja,gauri;
select books.title,customers.name from customers
join orders on orders.customer_id=customers.id
join books on orders.book_id=books.id
where customers.name in("rahul","vishal","rutuja","gauri") ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	name
DATA_SCIENCE_WITH_JUPYTER	RAHUL
DATA_SCIENCE_HANDBOOK	RUTUJA
INTRODUCING_DATA_SCIENCE	GAURI
DATA_SCIENCE_ON_GOOGLE_PLATFORM	VISHAL

 # 18. retrieve the name of book which is having lowest price.

select books.title,books.price from books
order by price asc limit 1;

```
# 24. retrieve the name of book which is having lowest price.
select books.title,books.price from books
order by price asc limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	price
DATA_SCIENCE_WITH_JUPYTER	539

 # 19. retrive the name of book which is having highest price.

select books.title,books.price from books
order by price desc limit 1;
-- or
select books.title,max(books.price) from books
group by books.title limit 1;

```
# 25. retrive the name of book which is having highest price.
select books.title,books.price from books
order by price desc limit 1;
-- or
select books.title,max(books.price) from books
group by books.title limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	price
DATA_SCIENCE_HANDBOOK	3553

 # 20. Retrieve the name of book which is having 3rd highest values.

select books.title,books.price from books
order by price desc limit 1 OFFSET 2 ;

```
# 26. Retrieve the name of book which is having 3rd highest values.
select books.title,books.price from books
order by price desc limit 1 OFFSET 2 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	price
ARTIFICIAL_INTELLIGENCE	1499

 # 21. retrieve the books details by increasing their values by rs.100 on each book.

select books.title,books.price as "old price",price+100 as "new price"
from books;

```
# 11. retrieve the books details by increasing their values by rs.100 on each book.
select books.title,books.price as "old price",price+100 as "new price"
from books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	old price	new price
DATA_SCIENCE_HANDBOOK	3553	3653
INTRODUCING_DATA_SCIENCE	835	935
MACHINE_LEARNING_USING_PYTHON	605	705
HAND_ON_DATA_SCIENCE	2619	2719
Data_Science_for_Business_Professionals	805	905
DATA_SCIENCE_WITH_JUPYTER	539	639
ARTIFICIAL_INTELLIGENCE	1499	1599
PYTHON_FOR_DATA_SCIENCE_DUMMIES	740	840
DATA_SCIENCE_ON_GOOGLE_PLATFORM	1225	1325

Result 10

 # 22. retrieve the total revenue collected before increasing price.

select sum(books.price) from books;

```
# 10. retrieve the total revenue collected before increasing price.
select sum(books.price) from books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(books.price)
12420

 # 23. retrieve total revenue profit after increasing price on book

select sum(books.price+100)-sum(books.price) from books;

```
# 12. retrieve total revenue profit after increasing price on book
select sum(books.price+100)-sum(books.price) from books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(books.price+100)-sum(books.price)
900

 # 24. retrieve the total revenue collected in terms of percentage after increasing price on a book

select ((sum(books.price+100)-sum(books.price))/(sum(books.price))*100) as "total_profit_gained_after_increasing_price" from books;

```
# 13. retrieve the total revenue collected in terms of percentage after increasing price on a book
select ((sum(books.price+100)-sum(books.price))/(sum(books.price))*100) as "total_profit_gained_after_increasing_price" from books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_profit_gained_after_increasing_price
7.2464

 # 25. notifying the future price increase in particular books

select books.title,books.price,price+100,
case books.title
when "ARTIFICIAL_INTELLIGENCE" then "price will be icrease by 150"

when "MACHINE_LEARNING_USING_PYTHON" then "price will be icrease by 200"
else "price = price +100"
end as "remarks" from books;

```
# 14. notifying the future price increase in particular books
select books.title,books.price,price+100,
case books.title
when "ARTIFICIAL_INTELLIGENCE" then "price will be increase by 150"
when "MACHINE_LEARNING_USING_PYTHON" then "price will be increase by 200"
else "price = price +100"
end as "remarks" from books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	price	price+100	remarks
DATA_SCIENCE_HANDBOOK	3553	3653	price = price +100
INTRODUCING_DATA_SCIENCE	835	935	price = price +100
MACHINE_LEARNING_USING_PYTHON	605	705	price will be increase by 200
HAND_ON_DATA_SCIENCE	2619	2719	price = price +100
Data_Science_for_Business_Professionals	805	905	price = price +100
DATA_SCIENCE_WITH_JUPYTER	539	639	price = price +100
ARTIFICIAL_INTELLIGENCE	1499	1599	price will be increase by 150
PYTHON_FOR_DATA_SCIENCE_DUMMIES	740	840	price = price +100
DATA_SCIENCE_ON_GOOGLE_PLATFORM	1225	1325	price = price +100

Result 15

26. List all the books that have been ordered.

SELECT DISTINCT books.id, books.title,books.price FROM books JOIN orders ON books.id = orders.book_id;

```
# 27. List all the books that have been ordered.
SELECT distinct(books.id), books.title,books.price FROM books JOIN orders ON books.id = orders.book_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	title	price
5	HAND_ON_DATA_SCIENCE	2619
4	MACHINE_LEARNING_USING_PYTHON	605
1	DATA_SCIENCE_HANDBOOK	3553
8	DATA_SCIENCE_WITH_JUPYTER	539
3	INTRODUCING_DATA_SCIENCE	835
10	PYTHON_FOR_DATA_SCIENCE_DUMMIES	740
11	DATA_SCIENCE_ON_GOOGLE_PLATFORM	1225
9	ARTIFICIAL_INTELLIGENCE	1499
6	Data_Science_for_Business_Professionals	805

Result 17

 # 27. retrieve the total number of customers who ordered book.

select count(customers.id) from customers
join orders on orders.customer_id=customers.id;

```
# 28. retrieve the total number of customers who ordered book.
select count(customers.id) from customers
join orders on orders.customer_id=customers.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(customers.id)
76

 # 28. Calculate the total number of orders made.

SELECT COUNT(id) AS total_orders FROM orders;

```
# 29. Calculate the total number of orders made.
SELECT COUNT(id) AS total_orders FROM orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_orders
78

 # 29. Calculate the total revenue generated by the bookstore.

SELECT SUM(price) AS total_revenue FROM books JOIN orders ON books.id = orders.book_id;

```
# 30. Calculate the total revenue generated by the bookstore.
SELECT SUM(price) AS total_revenue FROM books JOIN orders ON books.id = orders.book_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_revenue
126269

 # 30. List all the books that have been ordered more than once.

select books.id,books.title,count(orders.book_id) as total_book_ordered from books
join orders on orders.book_id=books.id
group by books.id,books.title having count(book_id)>1;

```
# 30. List all the books that have been ordered more than once.
select books.id,books.title,count(orders.book_id) as total_book_ordered from books
join orders on orders.book_id=books.id
group by books.id,books.title having count(book_id)>1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	title	total_book_ordered
5	HAND_ON_DATA_SCIENCE	12
4	MACHINE_LEARNING_USING_PYTHON	11
1	DATA_SCIENCE_HANDBOOK	13
8	DATA_SCIENCE_WITH_JUPYTER	4
3	INTRODUCING_DATA_SCIENCE	13
10	PYTHON_FOR_DATA_SCIENCE_DUMMIES	5
11	DATA_SCIENCE_ON_GOOGLE_PLATFORM	7
9	ARTIFICIAL_INTELLIGENCE	9
6	Data_Science_for_Business_Professionals	4

Result 1

 # 31. list all the books whose price is greater than 1000 and less than 3000

select books.title,books.price from books
where price>1000 and price<3000;

```
# 31. list all the books whose price is greater than 1000 and less than 3000
select books.title,books.price from books
where price>1000 and price<3000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	price
HAND_ON_DATA_SCIENCE	2619
ARTIFICIAL_INTELLIGENCE	1499
DATA_SCIENCE_ON_GOOGLE_PLATFORM	1225

 # 32. list all the books whose price is between 1000 and 3000

select books.title,books.price from books
where price between 1000 and 3000;

```
# 32. list all the books whose price is between 1000 and 3000
select books.title,books.price from books
where price between 1000 and 3000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	price
HAND_ON_DATA_SCIENCE	2619
ARTIFICIAL_INTELLIGENCE	1499
DATA_SCIENCE_ON_GOOGLE_PLATFORM	1225

 # 33. List the top 5 best-selling books.

SELECT books.id, books.title, COUNT(orders.book_id) AS num_orders FROM books JOIN orders ON books.id = orders.book_id GROUP BY books.id, books.title ORDER BY num_orders DESC LIMIT 5;

```
# 33. List the top 5 best-selling books.
SELECT books.id, books.title, COUNT(orders.book_id) AS num_orders FROM books
JOIN orders ON books.id = orders.book_id GROUP BY books.id, books.title ORDER BY num_orders DESC LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

id	title	num_orders
1	DATA_SCIENCE_HANDBOOK	13
3	INTRODUCING_DATA_SCIENCE	13
5	HAND_ON_DATA_SCIENCE	12
4	MACHINE_LEARNING_USING_PYTHON	11
9	ARTIFICIAL_INTELLIGENCE	9

 # 34. List the customers who have ordered a specific book by providing the book ID = 011.

SELECT customers.name,books.title,orders.purchase_date FROM customers
JOIN orders ON customers.id = orders.customer_id
join books ON orders.book_id=books.id WHERE orders.book_id = "011";

```
# 34. List the customers who have ordered a specific book by providing the book ID = 011.
SELECT customers.name,books.title,orders.purchase_date FROM customers
JOIN orders ON customers.id = orders.customer_id
join books ON orders.book_id=books.id WHERE orders.book_id = "011";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	title	purchase_date
VISHAL	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2022-02-05
GAURAV	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2022-08-22
GAURANG	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2021-01-23
NILKANTH	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2023-02-20
PANKAJ	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2023-02-20
PRANALI	DATA_SCIENCE_ON_GOOGLE_PLATFORM	2023-02-20

35. List the customers who have not made any orders.

SELECT id, name, email, registration_date FROM customers WHERE id not IN (SELECT customer_id FROM orders);

```
# 35. List the customers who have not made any orders.
SELECT id, name, email, registration_date FROM customers WHERE id not IN ( SELECT customer_id FROM orders );
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	email	registration_date
1457	SHUBHAM	shubham@gmail.com	2023-02-20
7474	KISHOR	KISHOR@gmail.com	2023-01-04
9085	ayush	yush@gmail.com	2019-08-03
9137	PRAKASH	PRAKASH@GMAIL.COM	2020-11-30
9785	abhay	abhay@gmail.com	2019-08-25
NULL	NULL	NULL	NULL

36. List the books that have not been ordered.

SELECT books.id, books.title, books.author FROM books WHERE books.id NOT IN (SELECT DISTINCT book_id FROM orders);

```
# 36. List the books that have not been ordered.
SELECT books.id, books.title, books.author FROM books WHERE books.id NOT IN ( SELECT DISTINCT book_id FROM orders );
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	author
NULL	NULL	NULL

37. Calculate the average price of all the books in the database.

SELECT AVG(price) AS avg_price FROM books;

```
# 37. Calculate the average price of all the books in the database.
SELECT AVG(price) AS avg_price FROM books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg_price
1380.0000

38. Calculate the total number of customers who have made an order.

SELECT COUNT(DISTINCT customer_id) AS num_customers FROM orders;

```
# 38. Calculate the total number of customers who have made an order.
SELECT COUNT(DISTINCT customer_id) AS num_customers FROM orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_customers
78

39. List the top 5 customers who have spent the most money.

SELECT customers.id, customers.name, SUM(books.price) AS total_spent FROM customers
JOIN orders ON customers.id = orders.customer_id
JOIN books ON orders.book_id = books.id GROUP BY customers.id, customers.name
ORDER BY total_spent DESC LIMIT 5;

```
# 39. List the top 5 customers who have spent the most money.
SELECT customers.id, customers.name, SUM(books.price) AS total_spent FROM customers
JOIN orders ON customers.id = orders.customer_id
JOIN books ON orders.book_id = books.id GROUP BY customers.id, customers.name ORDER BY total_spent DESC LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

id	name	total_spent
8574	AMRUTA	3553
1459	KUNAL	3553
8754	VISHWAJEET	3553
9898	RUSHIKESH	3553
7896	RUTUJA	3553

40. List the top 5 authors who have sold the most books.

SELECT books.author,books.title, COUNT(orders.book_id) AS num_books_sold FROM books
JOIN orders ON books.id = orders.book_id
join customers on orders.customer_id=customers.id
GROUP BY books.author,books.title ORDER BY num_books_sold DESC LIMIT 5;

```
# 40. List the top 5 authors who have sold the most books.
SELECT books.author,books.title, COUNT(orders.book_id) AS num_books_sold FROM books
JOIN orders ON books.id = orders.book_id
join customers on orders.customer_id=customers.id
GROUP BY books.author,books.title ORDER BY num_books_sold DESC LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

author	title	num_books_sold
FIELD_CADY	DATA_SCIENCE_HANDBOOK	13
DAVY CIELEN	INTRODUCING_DATA_SCIENCE	13
FRANKY_KANE	HAND_ON_DATA_SCIENCE	11
DR_THAREJA	MACHINE_LEARNING_USING_PYTHON	11
J_PAUL	ARTIFICIAL_INTELLIGENCE	9

41. List the customers who have ordered more than 3 books.

SELECT customers.id, customers.name, COUNT(orders.book_id) AS num_books_ordered FROM customers JOIN orders ON customers.id = orders.customer_id GROUP BY customers.id, customers.name HAVING COUNT(orders.book_id) > 1;

```
# 41. List the customers who have ordered more than 3 books.
SELECT customers.id, customers.name, COUNT(orders.book_id) AS num_books_ordered FROM customers
JOIN orders ON customers.id = orders.customer_id
GROUP BY customers.id, customers.name
HAVING COUNT(orders.book_id) > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

author	title	num_books_sold
FIELD_CADY	DATA_SCIENCE_HANDBOOK	13
DAVY CIELEN	INTRODUCING_DATA_SCIENCE	13
FRANKY_KANE	HAND_ON_DATA_SCIENCE	11
DR_THAREJA	MACHINE_LEARNING_USING_PYTHON	11
J_PAUL	ARTIFICIAL_INTELLIGENCE	9

42. List the books that have been ordered more than 10 times.

SELECT books.id, books.title, COUNT(orders.book_id) AS num_orders FROM books JOIN orders ON books.id = orders.book_id GROUP BY books.id, books.title HAVING COUNT(orders.book_id) > 10;

```
# 42. List the books that have been ordered more than 10 times.
SELECT books.id, books.title, COUNT(orders.book_id) AS num_orders FROM books
JOIN orders ON books.id = orders.book_id GROUP BY books.id, books.title HAVING COUNT(orders.book_id) > 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	title	num_orders
5	HAND_ON_DATA_SCIENCE	12
4	MACHINE_LEARNING_USING_PYTHON	11
1	DATA_SCIENCE_HANDBOOK	13
3	INTRODUCING_DATA_SCIENCE	13

43. List the customers who have ordered a book with a price higher than the average book price.

SELECT customers.id, customers.name, books.title, books.price FROM customers JOIN orders ON customers.id = orders.customer_id
JOIN books ON orders.book_id = books.id
WHERE books.price > (SELECT AVG(price) FROM books);

```
# 43. List the customers who have ordered a book with a price higher than the average book price.
SELECT customers.id, customers.name, books.title, books.price FROM customers
JOIN orders ON customers.id = orders.customer_id
JOIN books ON orders.book_id = books.id
WHERE books.price > ( SELECT AVG(price) FROM books );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	title	price
1999	ADITYA	HAND_ON_DATA_SCIENCE	2619
1459	KUNAL	DATA_SCIENCE_HANDBOOK	3553
7896	RUTUJA	DATA_SCIENCE_HANDBOOK	3553
4578	APURVA	ARTIFICIAL_INTELLIGENCE	1499
7555	RAM	ARTIFICIAL_INTELLIGENCE	1499
4488	RAGHAV	ARTIFICIAL_INTELLIGENCE	1499
7485	SONALI	ARTIFICIAL_INTELLIGENCE	1499
8574	AMRUTA	DATA_SCIENCE_HANDBOOK	3553
8754	VISHWAJEET	DATA_SCIENCE_HANDBOOK	3553
9898	RUSHIKESH	DATA_SCIENCE_HANDBOOK	3553

Result 13

Output

44. List the books that have not been ordered.

SELECT id, title FROM books WHERE id NOT IN (SELECT book_id FROM orders);

```
# 44. List the books that have not been ordered.
SELECT id, title FROM books WHERE id NOT IN ( SELECT book_id FROM orders );
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title
NULL	NULL

45. List the authors who have never sold any books.

SELECT author FROM books WHERE author NOT IN (SELECT author FROM books JOIN orders ON books.id = orders.book_id GROUP BY author);

```
# 45. List the authors who have never sold any books.
SELECT author FROM books WHERE author NOT IN ( SELECT author FROM books JOIN orders ON books.id = orders.book_id GROUP BY author );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

author

46. Calculate the total revenue generated by the bookstore per author.

SELECT books.author, SUM(books.price) AS total_revenue FROM books JOIN orders ON books.id = orders.book_id GROUP BY books.author;

```
# 45. List the authors who have never sold any books.
SELECT author FROM books WHERE author NOT IN ( SELECT author FROM books JOIN orders ON books.id = orders.book_id GROUP BY author );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

author

47. Calculate the average number of books ordered per customer.

SELECT AVG(num_books_ordered) AS avg_num_books_ordered FROM (SELECT customer_id, COUNT(book_id) AS num_books_ordered FROM orders GROUP BY customer_id) subquery;

```
# 47. Calculate the average number of books ordered per customer.
SELECT AVG(num_books_ordered) AS avg_num_books_ordered FROM ( SELECT customer_id, COUNT(book_id) AS num_books_ordered FROM orders GROUP BY customer_id ) subquery;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg_num_books_ordered
1.0000