Pune Institute of Computer Technology, Pune

Department of Computer Engineering

A.Y. 2020-21 Semester: I

Database Management System Lab

Name: Aditya Sawant

Roll No: 31302

Batch: K3

ASSIGNMENT 4

TITLE:

Design at least 10 SQL Queries for suitable database application using MYSQL (All types of Join , Sub-Query and View)

PROBLEM DEFINITION:

Execute the following queries:

1. Find the professor details and department details using NATURAL JOIN.

2. Find the prof\_id, prof\_name and shift.

3. List all the department details and the corresponding names of professors in the same department.

4. List all the professors and the corresponding names of department.

5. Display professor name, dept\_name, shift, salary where prof\_id = 101;

6. list the total number of professor in each department.

7. List the prof\_id associated department and the dept\_name having name ‘computer’;

8. Find the names of all departments where the professors joined in year 2015 (or date of joining is 1-1-2015). 9. Create view showing the professor and shift details.

10. Perform Manipulation on simple view-Insert, update, delete, drop view.

OBJECTIVE:

To understand

* Types of joins,
* Subquery and its types,
* Complex views

OUTCOME:

We will be able to understand types of joins , subquery and its types and complex views and implement MYSQL Queries for suitable database application

HARDWARE REQUIREMENTS:

* MONITOR
* KEYBOARD
* 2GB RAM
* 2.4GHz I5 PROCESSOR

SOFTWARE REQUIREMENTS:

* DATABASE-MYSQL
* OS-FEDORA 20

THEORY:

JOINS:

A join is a method of linking data between one ([self-join](http://www.mysqltutorial.org/mysql-self-join/)) or more tables based on values of the common column between the tables.

TYPES OF JOINS:

MySQL supports the following types of joins:

1. NATURAL JOIN
2. INNER JOIN
3. LEFT JOIN
4. RIGHT JOIN
5. CROSS JOIN

NATURAL JOIN:

Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.

SYNTAX-

SELECT \* FROM table1 NATURAL JOIN table2;

INNER JOIN:

The INNER JOIN matches each row in one table with every row in other tables and allows you to query rows that contain columns from both tables.

The INNER JOIN is an optional clause of the SELECT statement. It appears immediately after the FROM clause. Here is the syntax of the INNER JOIN clause:

SYNTAX-

SELECT \* FROM table1 INNER JOIN table2 ON join\_condition1 INNER JOIN table3 ON join\_condition2 ...;

LEFT JOIN:

The LEFT JOIN allows you to query data from two or more tables. Similar to the INNER JOIN clause, the LEFT JOIN is an optional clause of the SELECT statement, which appears immediately after the FROM clause.

SYNTAX-

SELECT \* FROM table1 LEFT JOIN table2 ON join\_condition;

RIGHT JOIN:

MySQL RIGHT JOIN is similar to LEFT JOIN, except that the treatment of the joined tables is reversed.

SYNTAX-

SELECT \* FROM table1 RIGHT JOIN table2 ON join\_condition;

CROSS JOIN:

The CROSS JOIN clause returns the Cartesian product of rows from the joined tables. The result set will include all rows from both tables, where each row is the combination of the row in the first table with the row in the second table. In general, if each table has n and m rows respectively, the result set will have n x m rows.

SYNTAX-

SELECT \* FROM table1 CROSS JOIN table2 ;

SUB-QUERY:

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

VIEWS:

Views are virtual tables that do not store any data of their own but display data stored in other tables. In other words, VIEWS are nothing but SQL Queries. A view can contain all or a few rows from a table. A MySQL view can show data from one table or many tables.

SYNTAX-

CREATE VIEW view\_name AS SELECT STATEMENT;

### Types of views:

1. Simple View-

Simple view is view created on single table. When user wants data or some columns from same table then simple view is used.For an example if there is employee table which has Employee\_num,Employee\_name,salary columnns and we just need to see Employee\_num,Employee\_name then user can create a simple view.

1. Complex View-

Complex view is view created on more than 1 tables. Complex view is created on using more than one tables.When user wants to retrieve data from more than 1 table then we have to use complex views.To create complex view there should be relation between 2 tables else cartesian product will come by joining 2 tables .There should be some joining conditions,some filters needs to be considered while creating complex views.

PURPOSE OF VIEWS:

* Security-

You can restrict users to access directly to a table and allow them to access a subset of data via views. For example, you can allow users to access customer name, phone, email via a view but restrict them to access the bank account and other sensitive information.

* Simplicity-

A relational database may have many tables with complex relationships e.g., one-to-one and one-to-many that make it difficult to navigate. However, you can simplify the complex queries with joins and conditions using a set of views.

* Consistency-

Sometimes, you need to write a complex formula or logic in every query. To make it consistent, you can hide the complex queries logic and calculations in views. Once views are defined, you can reference the logic from the views rather than rewriting it in separate queries.

COMMANDS OF VIEWS:

* CREATE VIEW-

CREATE VIEW view\_name AS SELECT column1, column2, ... FROM table\_name WHERE condition;

* INSERT IN VIEW-

INSERT INTO view\_name ( field1, field2,...fieldN )VALUES(value1, value2,...valueN );

* UPDATE VIEW-

UPDATE view\_name AS SELECT column1, column2, ... FROM table\_name WHERE condition;

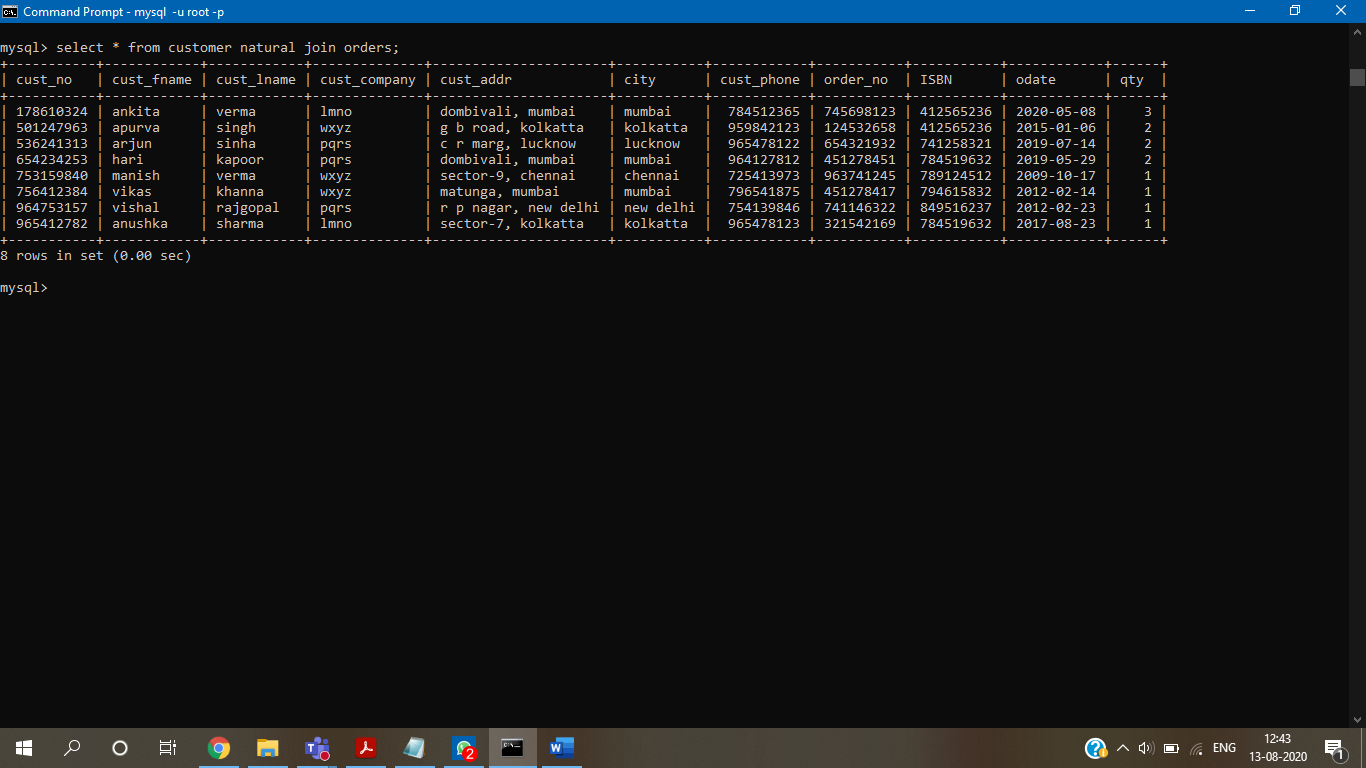
* DELETE FROM VIEW-

DELETE view\_name AS SELECT column1, column2, ... FROM table\_name WHERE condition;

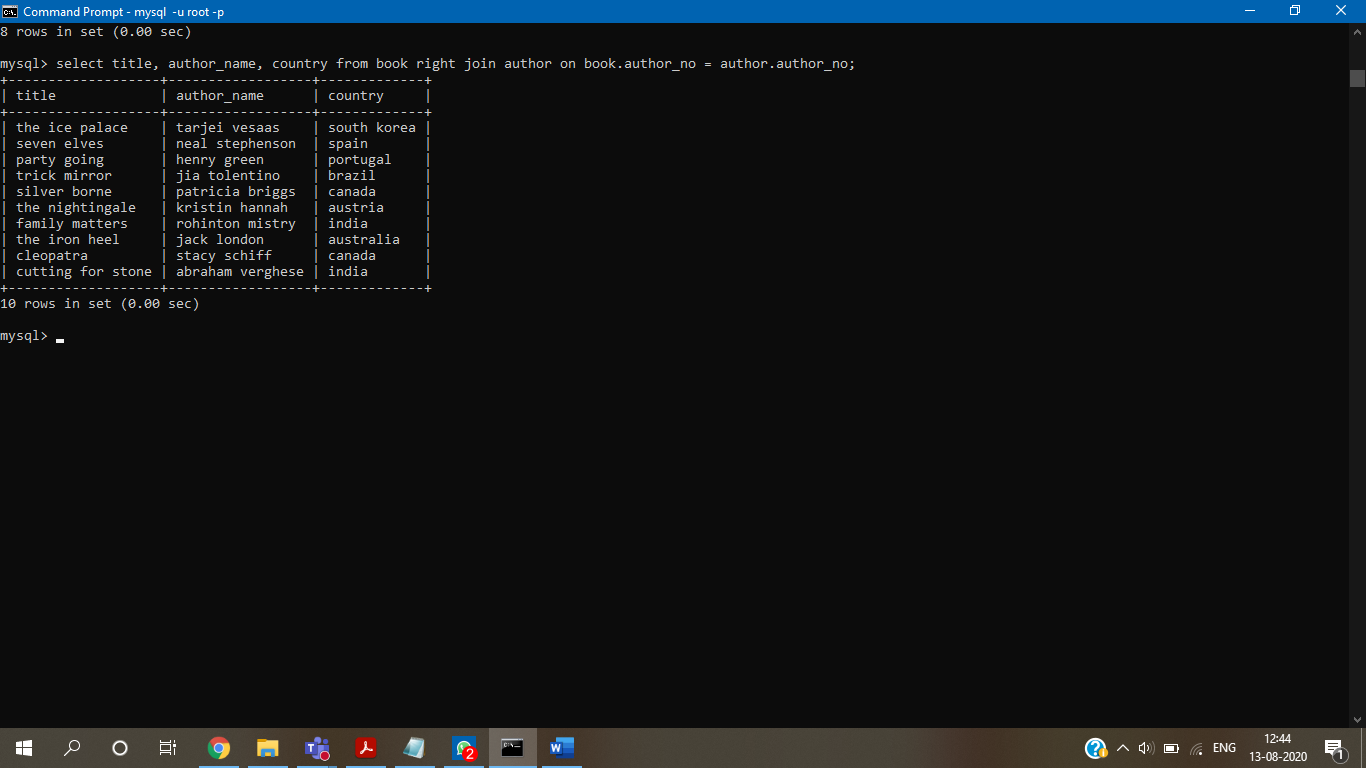
* DROP VIEW-

DROP VIEW view\_name;

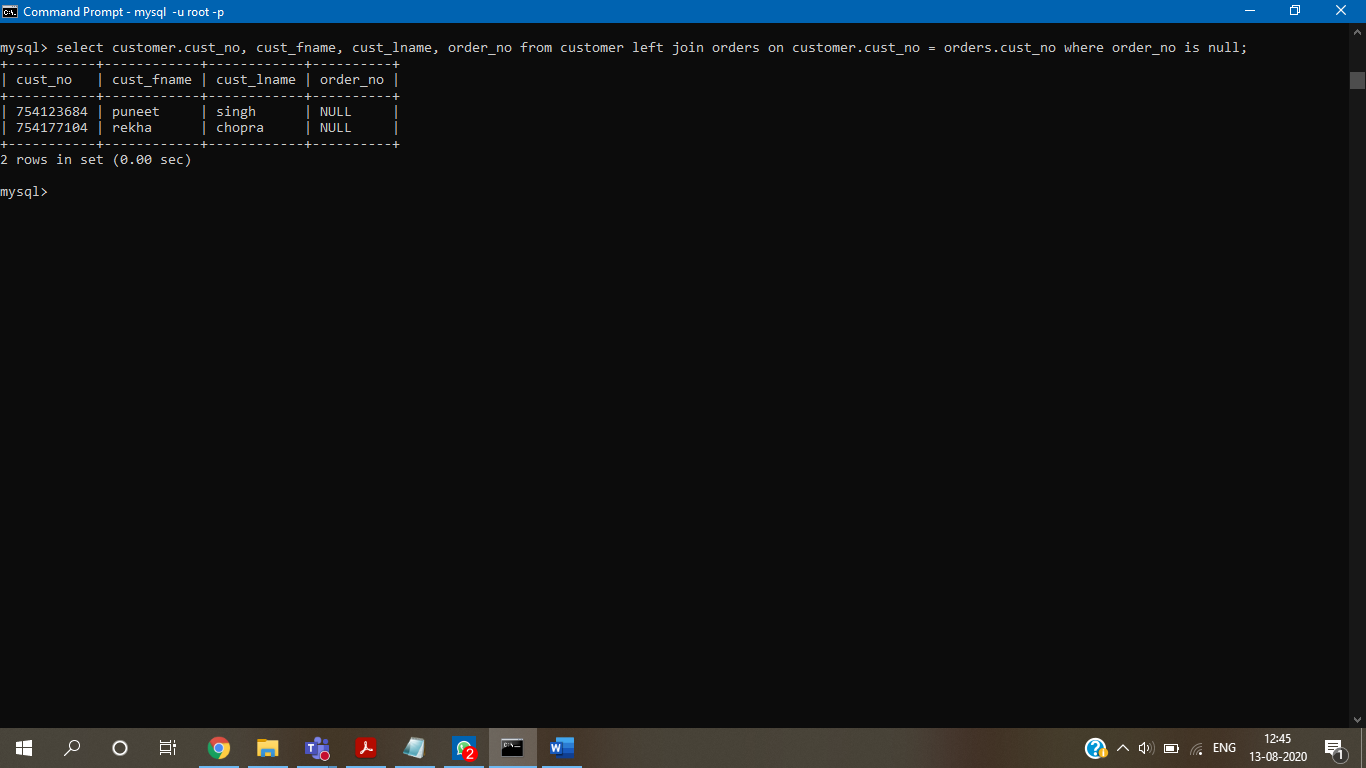
Find Customer details and order details using NATURAL JOIN.



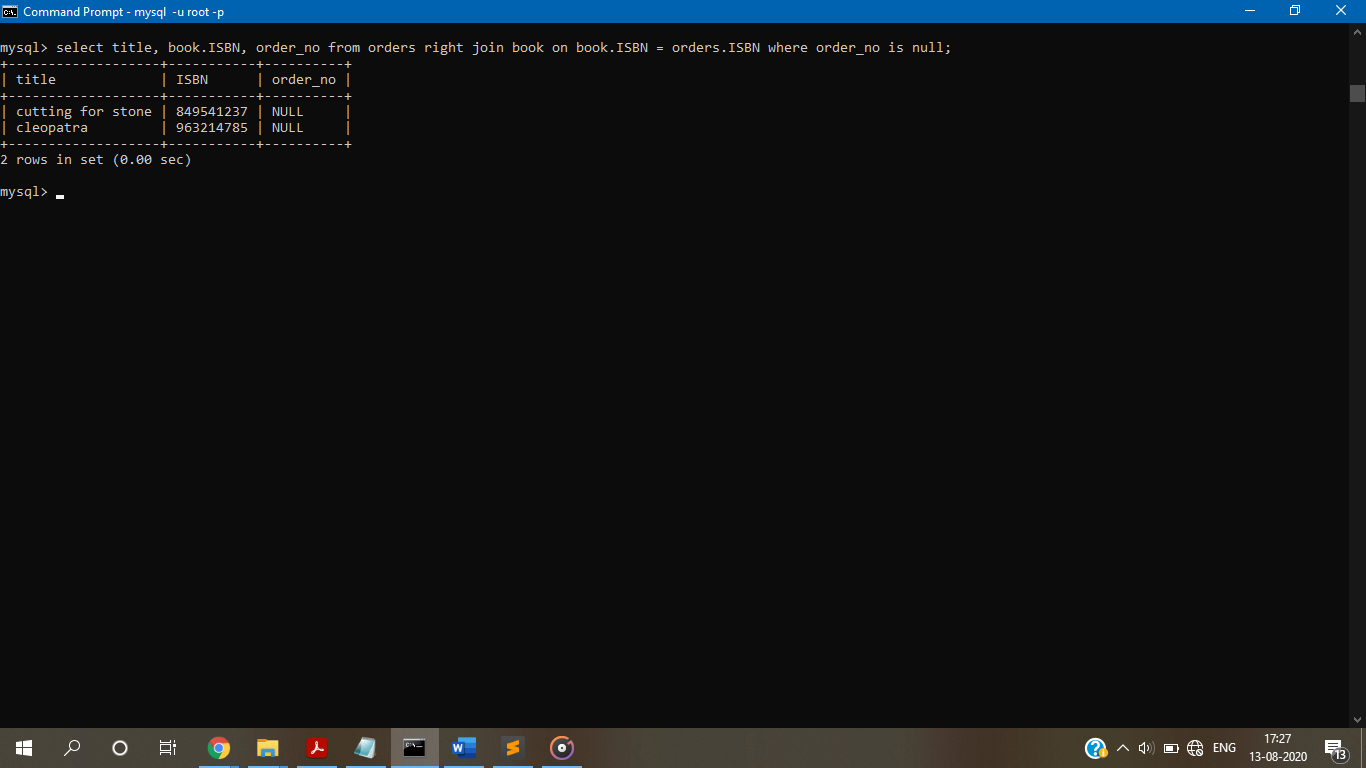
Find the book\_ title, author\_name, country.



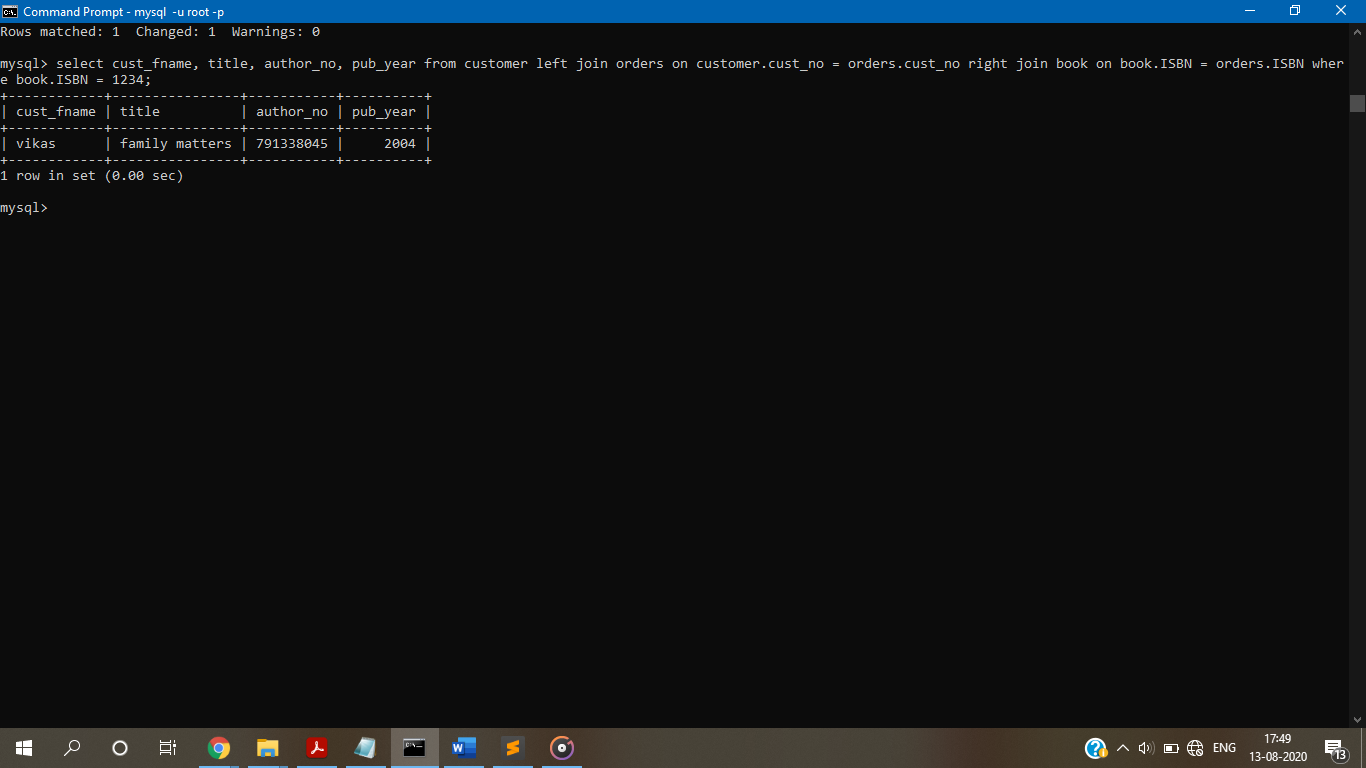
Find the customer ID, name and order\_no of customers who have never placed an order.



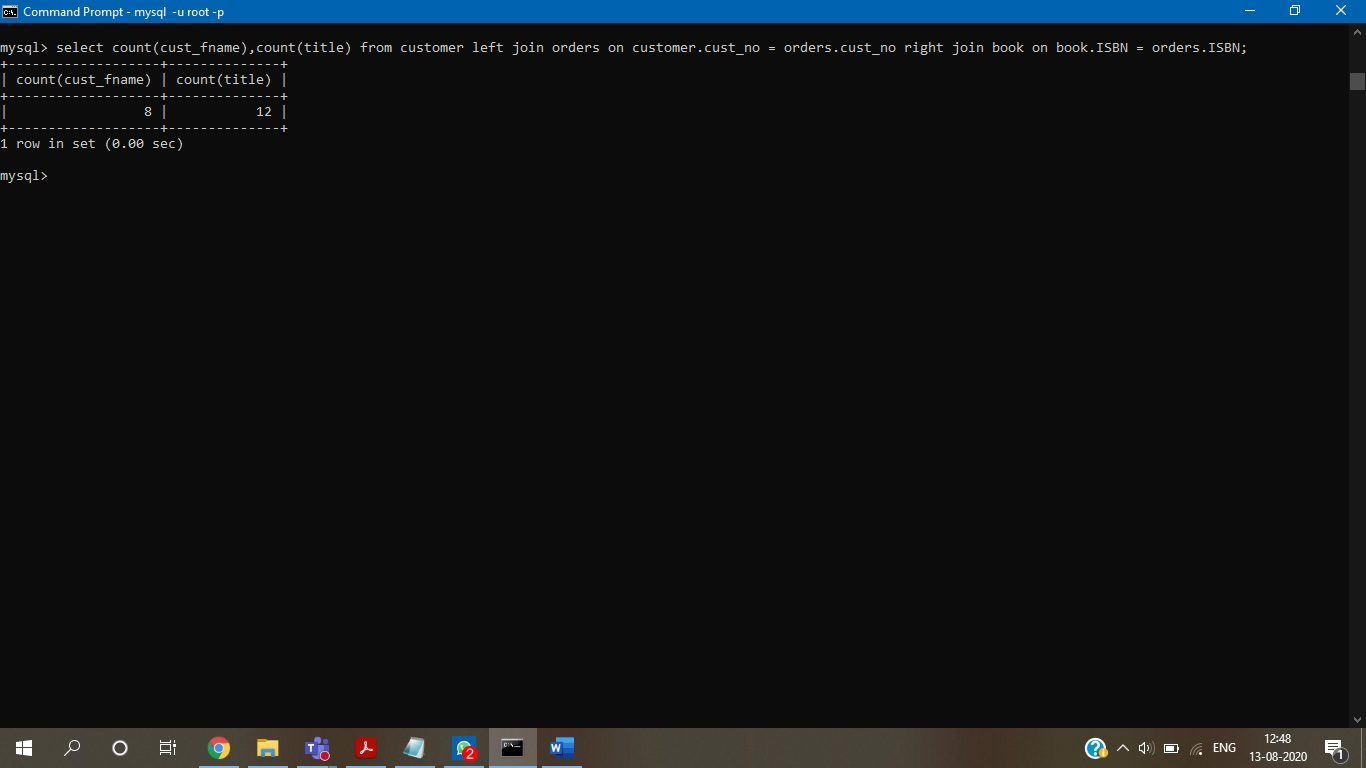
Find the Title, ISBN, order\_no of the books for which order is not placed.



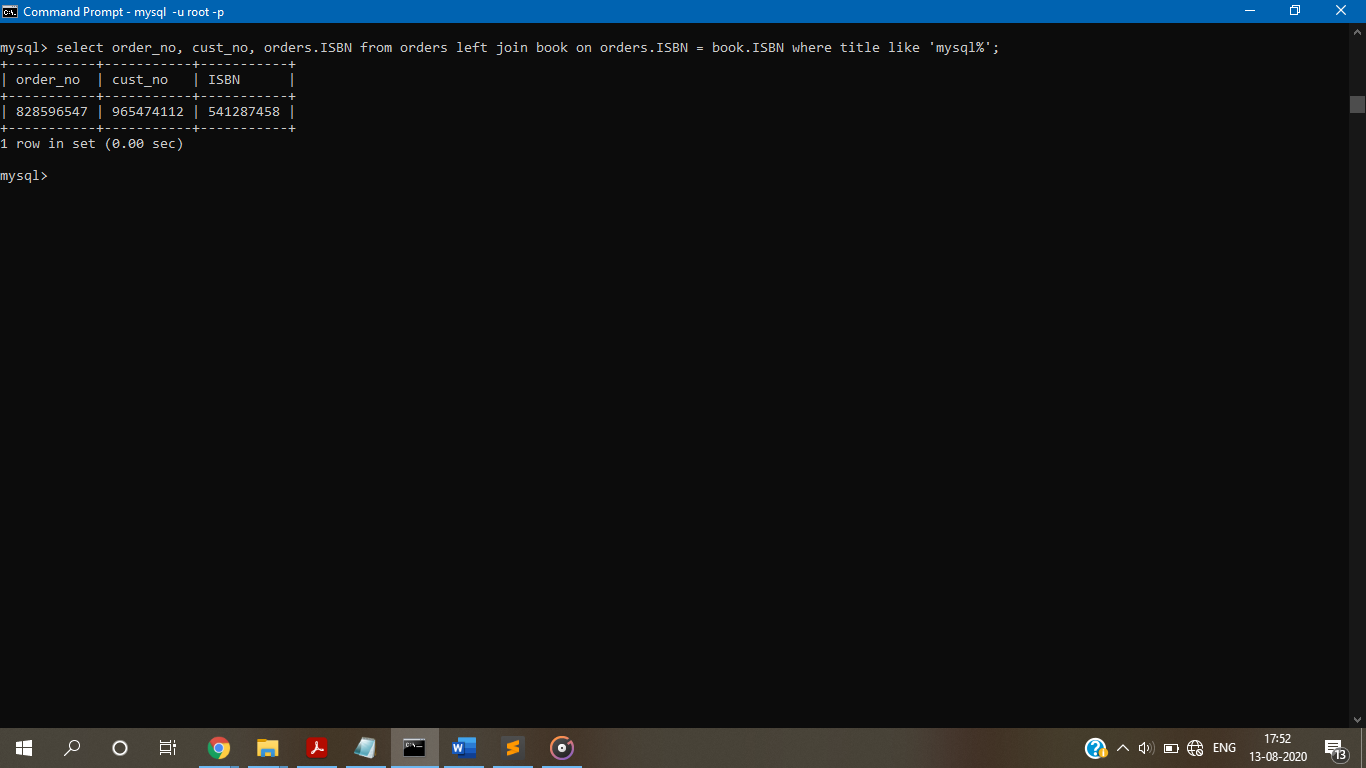
Display cust\_fname, title,author\_no,publisher\_year where ISBN=1234.



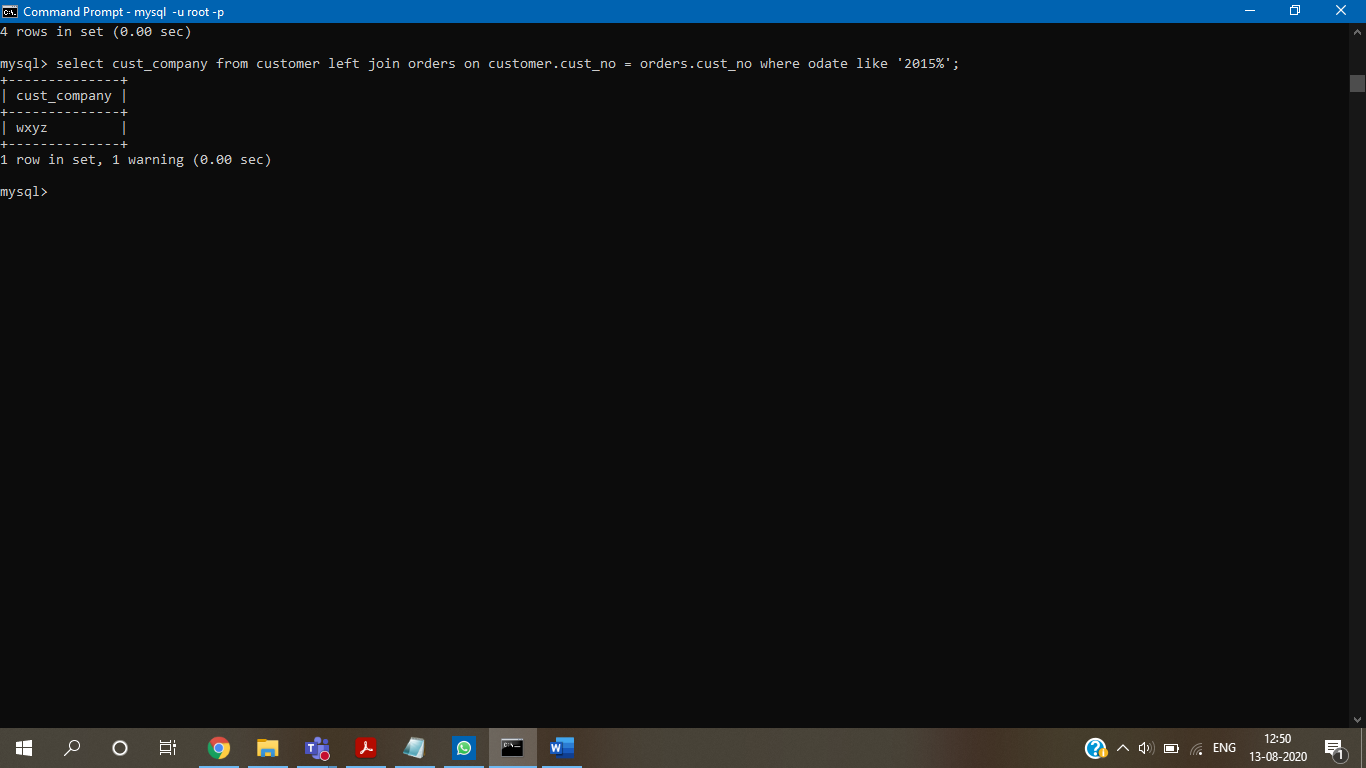
Display the total number of books and customer name.



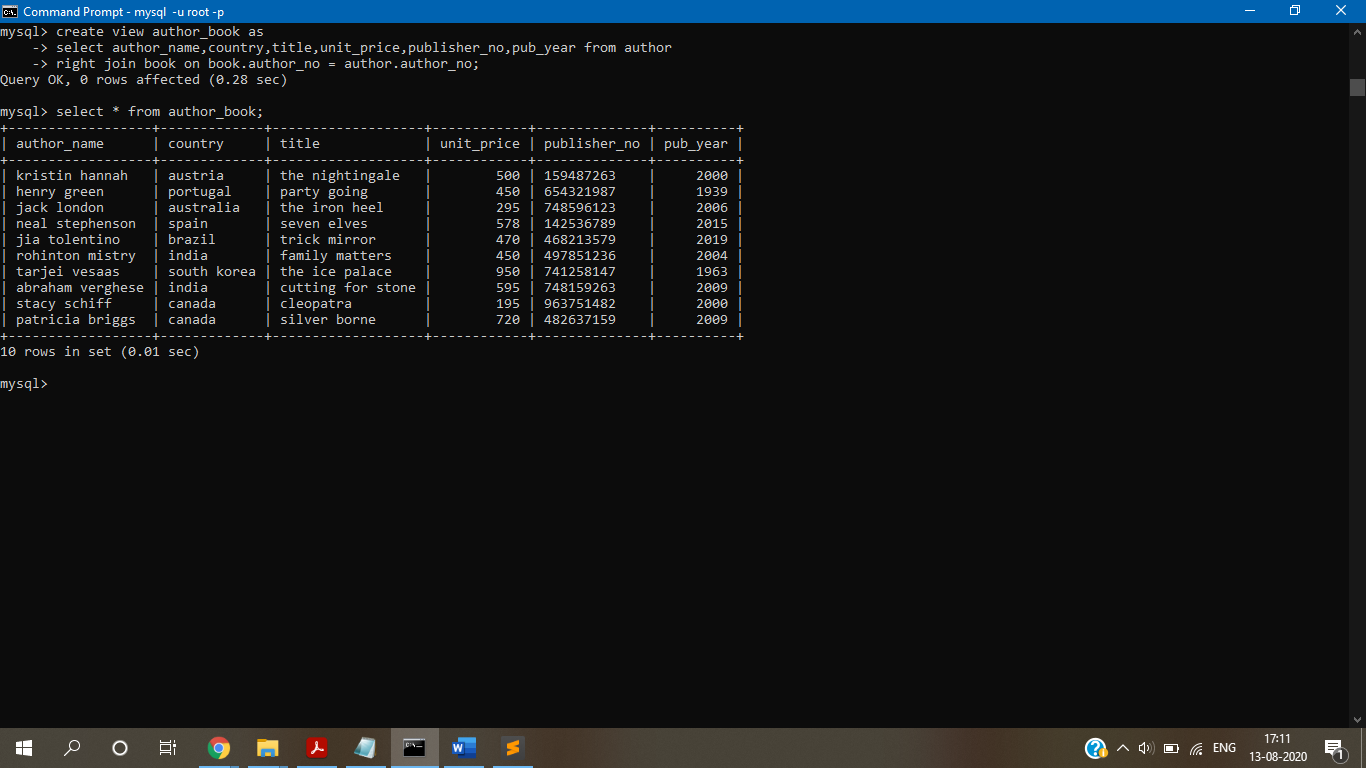
List the cust\_id, order\_no and ISBN with books having title 'mysql'.

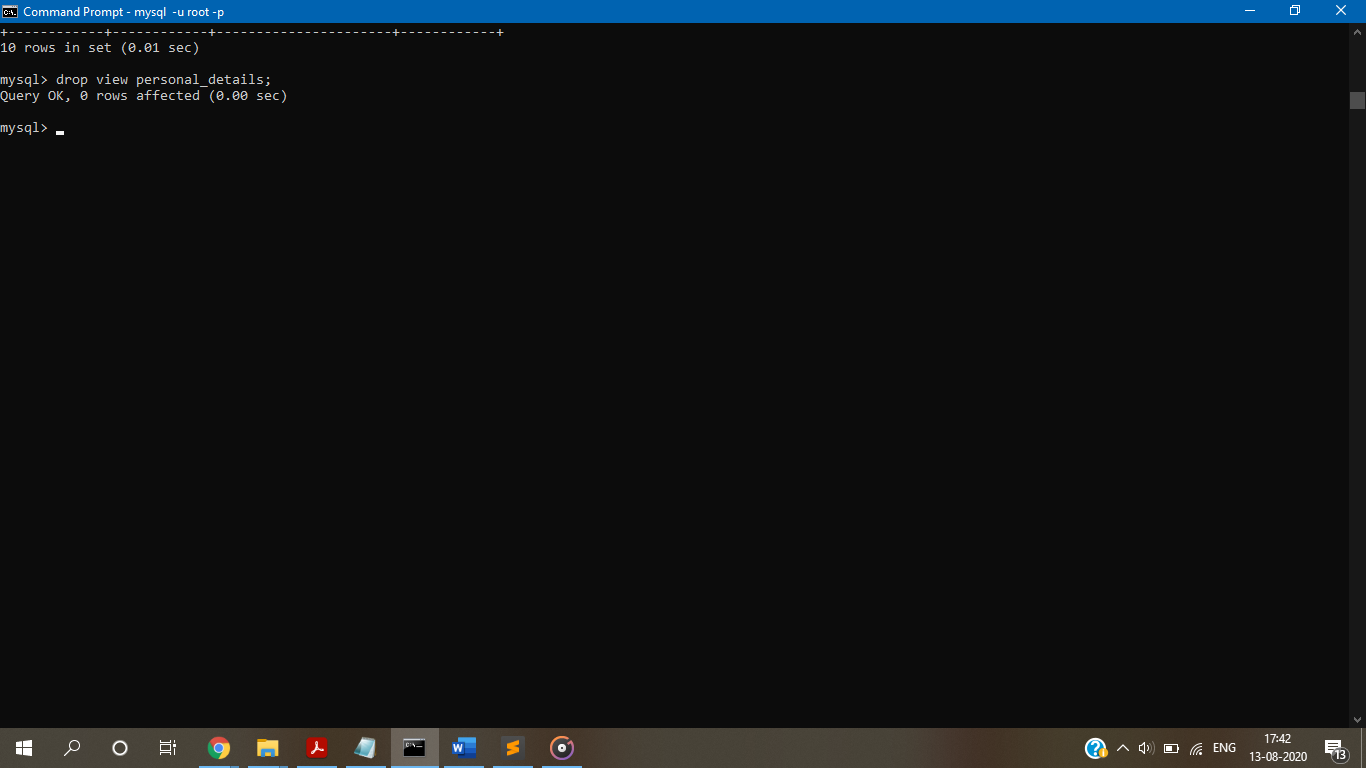
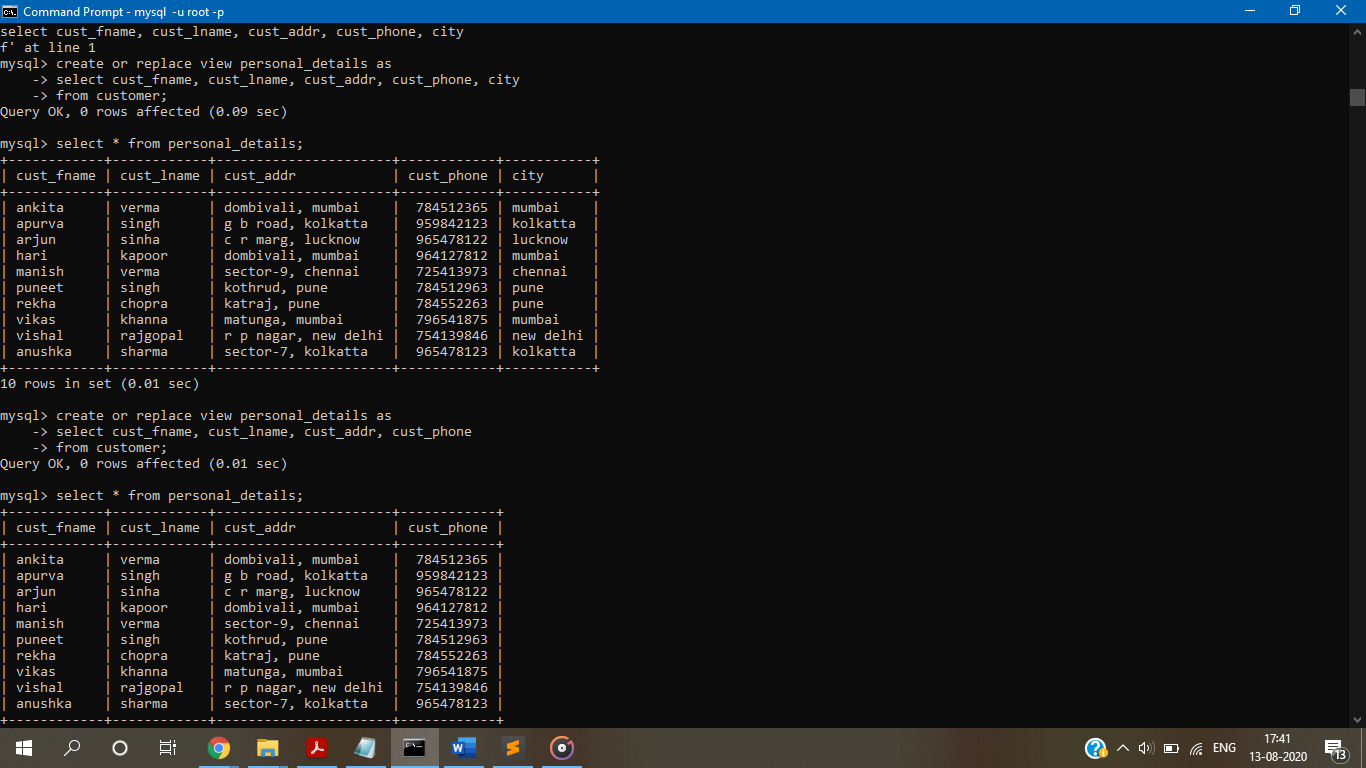
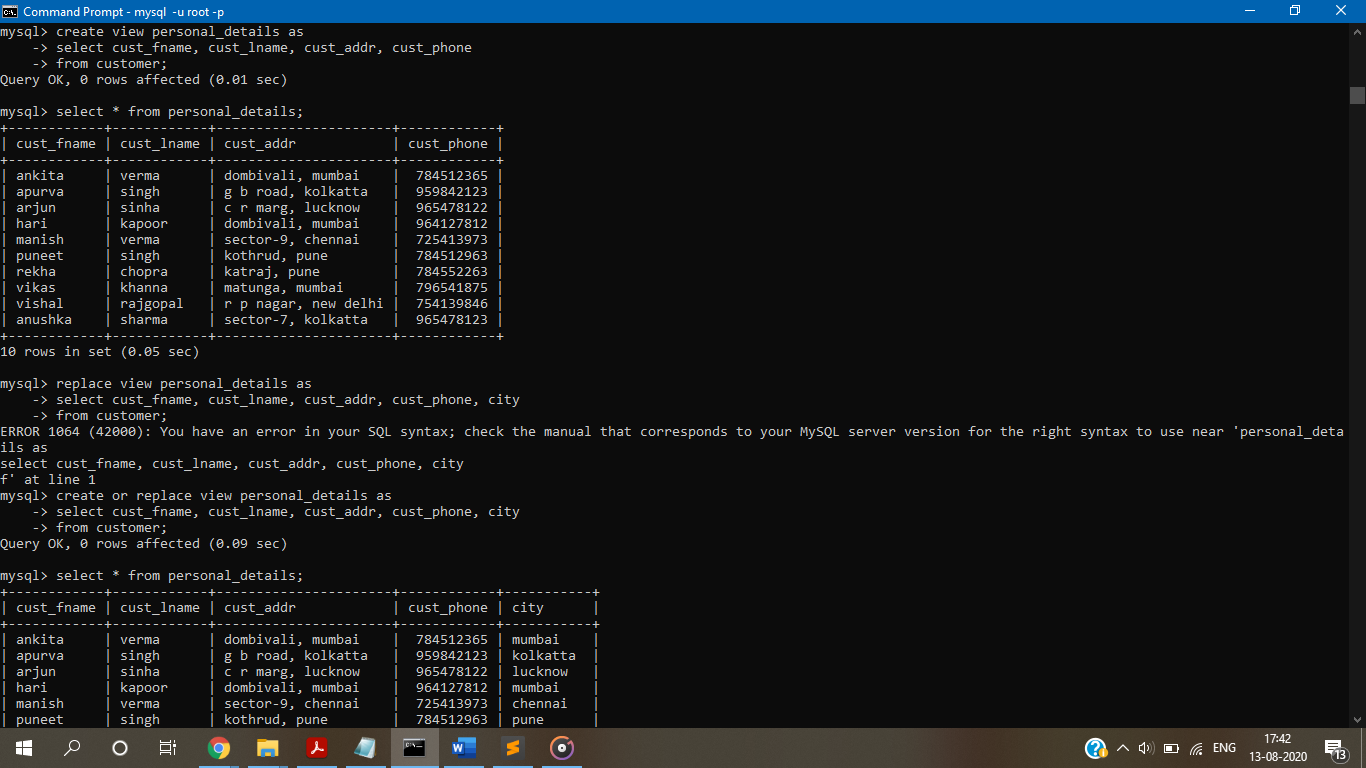


Find the names of all the companies that ordered books in the year 2015.



Create view showing the author and book details.



Perform Manipulation on simple view-Insert, update, delete, drop view. 

CONCLUSION:-

We successfully understood types of joins , subquery and its types and complex views and implemented MYSQL Queries for suitable database application.