MODULE:5 (DATABASE)

Q-1 What do you understand By Database

A database is a collection of data that is organized, which is also called structured data. It can be accessed or stored in a computer system. It can be managed through a Database management system (DBMS), a software used to manage data.

Q-2 What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Q-3 What is Difference between DBMS and RDBMS?

DBMS	RDBMS
DBMS stores data as file.	RDBMS stores data in tabular form.
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other.
Normalization is not present.	Normalization is present.
DBMS does not support distributed database.	RDBMS supports distributed database.
It deals with small quantity of data.	It deals with large amount of data.

DBMS	RDBMS
It is used for small organization and deal with small data.	It is used to handle large amount of data.
Security is less	More security measures provided.
It supports single user.	It supports multiple users.
Examples: XML, Window Registry, Forxpro, dbaseIIIplus etc.	Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

Q-4 What is MF Cod Rule of RDBMS Systems?

- **Information Rule**: All data is stored in tables.
- **Guaranteed Access Rule**: Each piece of data can be accessed using the table name, a primary key, and the column name.
- **Systematic Treatment of Null Values**: Null values are used to represent missing or unknown data in a consistent way.
- **Dynamic Online Catalog**: The database's structure and metadata are stored in the same way as regular data and can be queried.
- **Comprehensive Data Language**: The database must support a language that can handle data definition, manipulation, constraints, and transactions.
- **View Updating Rule**: Any view that can theoretically be updated should be updatable by the system.
- **High-Level Insert, Update, and Delete**: The database should support operations that handle multiple rows of data at once.
- **Physical Data Independence**: Changes to how data is stored should not affect how it is accessed.
- **Logical Data Independence**: Changes to the logical structure of the database (like adding new fields) should not affect applications.
- **Integrity Independence**: Integrity constraints (like primary keys) should be stored in the database and not in application programs.
- **Distribution Independence**: The database should work the same way even if distributed across different locations.

- **Non-Subversion Rule**: Low-level access methods cannot bypass integrity rules defined by the database.

Q-5 What do you understand By Data Redundancy?

Data redundancy is the unnecessary duplication of data in a database. It leads to increased storage costs, data inconsistency, maintenance challenges, and performance issues. It can be avoided through normalization and the use of foreign keys.

Q-6 What is DDL Interpreter?

A DDL Interpreter processes Data Definition Language (DDL) statements in a database management system to create, modify, and delete database objects like tables and indexes.

DDL commands come in the following types: CREATE, ALTER, DROP, RENAME, and TRUNCATE.

DDL statements only modify the database's schema; they have no direct effect on the data within the database.

DDL declarations are irreversible and difficult to undo.

Q-7 What is DML Compiler in SQL?

A DML (Data Manipulation Language) Compiler in SQL translates DML statements, such as SELECT, INSERT, UPDATE, and DELETE, into low-level instructions that the database management system can execute to manipulate data stored in the database.

Q-8 What is SQL Key Constraints writing an Example of SQL Key Constraints

SQL key constraints are rules applied to columns in a database table to ensure the integrity and uniqueness of the data. Common key constraints include:

1. **Primary Key**: Ensures each row in a table is unique and not null.
2. **Foreign Key**: Enforces a link between the data in two tables.
3. **Unique Key**: Ensures all values in a column are unique.
4. **Check Constraint**: Ensures that all values in a column satisfy a specific condition.
5. **Not Null**: Ensures that a column cannot have a null value.

Example of SQL Key Constraints:

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   email VARCHAR(255) UNIQUE,
   department_id INT,
   salary DECIMAL(10, 2) CHECK (salary > 0),
   manager_id INT,
   FOREIGN KEY (department_id) REFERENCES departments(department_id),
   FOREIGN KEY (manager_id) REFERENCES employees(employee_id),
   name VARCHAR(100) NOT NULL
);
```

Q-9 What is save Point? How to create a save Point write a Query?

- Savepoint is a command in SQL that is used with the rollback command.
- It is a command in Transaction Control Language that is used to mark the transaction in a table.
- Consider you are making a very long table, and you want to roll back only to a certain position in a table then; this can be achieved using the savepoint.
- If you made a transaction in a table, you could mark the transaction as a certain name, and later on, if you want to roll back to that point, you can do it easily by using the transaction's name.
- Savepoint is helpful when we want to roll back only a small part of a table and not the whole table. In simple words, we can say savepoint is a bookmark in SQL.

```
START TRANSACTION;
INSERT INTO employees (employee_id, name, department_id, salary)
VALUES (1, 'John Doe', 10, 50000);
SAVEPOINT savepoint1;
```

```
INSERT INTO departments (department_id, department_name)

VALUES (10, 'HR');

ROLLBACK TO savepoint1;

INSERT INTO employees (employee_id, name, department_id, salary)

VALUES (2, 'Jane Smith', 20, 60000);

COMMIT;
```

Q-10 What is trigger and how to create a Trigger in SQL?

In SQL, a trigger is a special type of stored procedure that automatically executes when certain events occur in the database. These events can include inserting, updating, or deleting data from a table. Triggers are useful for enforcing business rules, maintaining data integrity, and automating tasks based on database events.

```
delimiter //
create trigger loginsert
after insert
on student
for each row
begin
insert into logdata(log_action,update_date)values("record inserted!",now());
end //
delimiter ;
```

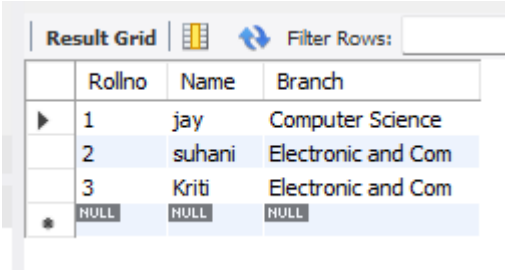
Task 1. Create Table Name : Student and Exam

Query:

```
create table student(Rollno int primary key auto_increment,Name varchar(20),Branch varchar(40));

insert into student(Name,Branch) values
('jay', 'Computer Science'),
('suhani', 'Electronic and Com'),
```

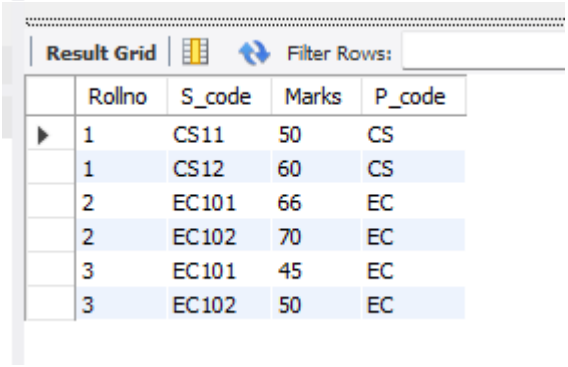
```
('Kriti', 'Electronic and Com');
select*from student;

create table exam( Rollno int,
   S_code varchar(50), Marks int, P_code varchar(50),
   foreign key  (Rollno) references student(Rollno));
insert into exam (Rollno, S_code, Marks, P_code) values
(1, 'CS11', 50, 'CS'),
(1, 'CS12', 60, 'CS'),
(2, 'EC101', 66, 'EC'),
(2, 'EC102', 70, 'EC'),
(3, 'EC101', 45, 'EC'),
(3, 'EC102', 50, 'EC');
select*from exam;
```

output:



Result Grid | Filter Rows:

Rollno	Name	Branch
1	jay	Computer Science
2	suhani	Electronic and Com
3	Kriti	Electronic and Com
NULL	NULL	NULL



Result Grid | Filter Rows:

Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

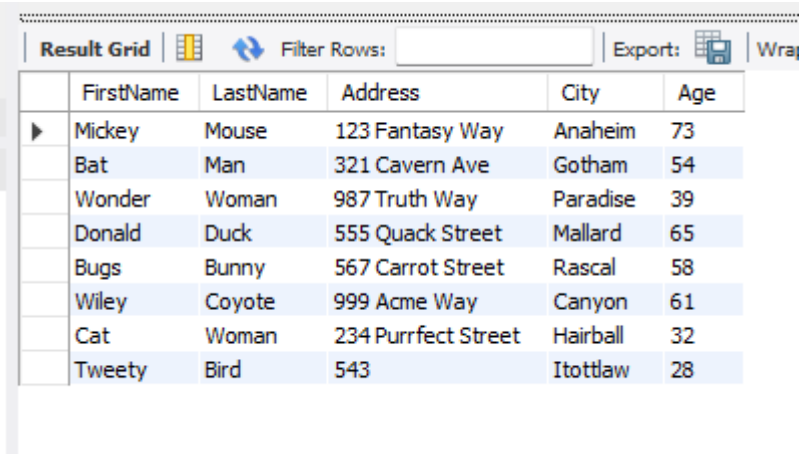
2 . Create table given below

Query:

```
create table people (
   FirstName VARCHAR(20),LastName varchar(20),  Address varchar(100),
   City varchar(50),Age int);
```

```
insert into people (FirstName, LastName, Address, City, Age) values
('Mickey', 'Mouse', '123 Fantasy Way', 'Anaheim', 73),
('Bat', 'Man', '321 Cavern Ave', 'Gotham', 54),
('Wonder', 'Woman', '987 Truth Way', 'Paradise', 39),
('Donald', 'Duck', '555 Quack Street', 'Mallard', 65),
('Bugs', 'Bunny', '567 Carrot Street', 'Rascal', 58),
('Wiley', 'Coyote', '999 Acme Way', 'Canyon', 61),
('Cat', 'Woman', '234 Purrfect Street', 'Hairball', 32),
('Tweety', 'Bird', '543', 'Itottlaw', 28);

select*from people;
```

output:



Result Grid | Filter Rows: | Export:

FirstName	LastName	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itottlaw	28

3. Create table given below: Employee and Incentive

Table Name: Employee

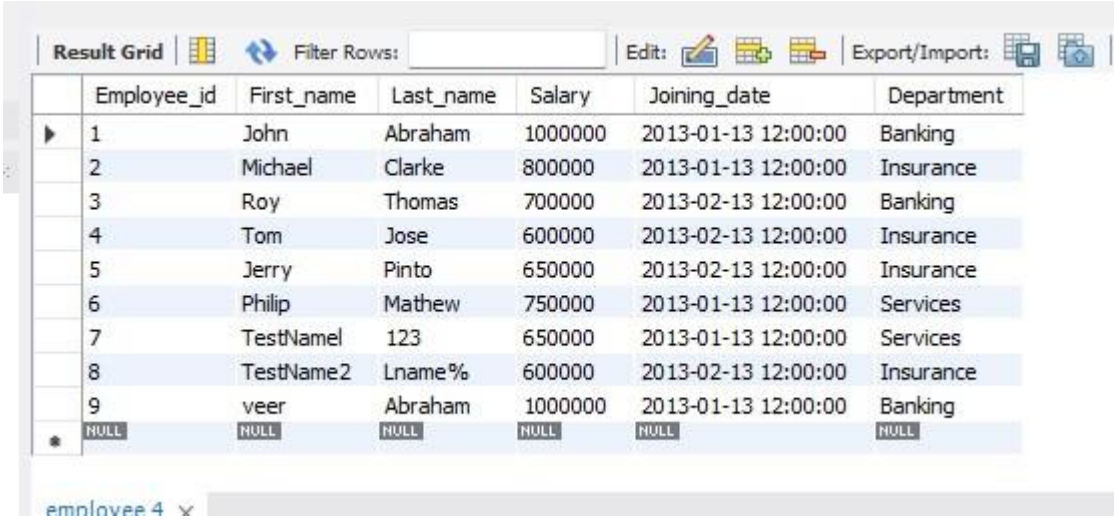
Query:

```
create table Employee (
 Employee_id INT primary key auto_increment, First_name VARCHAR(50),
 Last_name VARCHAR(50), Salary INT,  Joining_date datetime,
 Department VARCHAR(50)
);

insert into  Employee (First_name, Last_name, Salary, Joining_date,
Department) values
('John', 'Abraham', 1000000, '2013-01-13 12.00.00', 'Banking'),
('Michael', 'Clarke', 800000, '2013-01-13 12.00.00', 'Insurance'),
```

('Roy', 'Thomas', 700000, '2013-02-13 12.00.00', 'Banking'),
('Tom', 'Jose', 600000, '2013-02-13 12.00.00', 'Insurance'),
('Jerry', 'Pinto', 650000, '2013-02-13 12.00.00 ', 'Insurance'),
('Philip', 'Mathew', 750000, '2013-01-13 12.00.00', 'Services'),
('TestNamel', '123', 650000, '2013-01-13 12.00.00', 'Services'),
('TestName2', 'Lname%', 600000, '2013-02-13 12.00.00', 'Insurance');

select*from employee;

output:



Result Grid | Filter Rows: | Edit: | Export/Import:

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-13 12:00:00	Banking
2	Michael	Clarke	800000	2013-01-13 12:00:00	Insurance
3	Roy	Thomas	700000	2013-02-13 12:00:00	Banking
4	Tom	Jose	600000	2013-02-13 12:00:00	Insurance
5	Jerry	Pinto	650000	2013-02-13 12:00:00	Insurance
6	Philip	Mathew	750000	2013-01-13 12:00:00	Services
7	TestNamel	123	650000	2013-01-13 12:00:00	Services
8	TestName2	Lname%	600000	2013-02-13 12:00:00	Insurance
9	veer	Abraham	1000000	2013-01-13 12:00:00	Banking
NULL	NULL	NULL	NULL	NULL	NULL

employee 4

Table Name: Incentive

Query:

create table Incentive(Employee_ref_id int ,Incentive_date date ,Incentive_amount int,foreign key (Employee_ref_id)references Employee(Employee_id));

insert into Incentive (Employee_ref_id, Incentive_date, Incentive_amount) VALUES
(1, '2013-02-01', 5000),(2, '2013-02-01', 3000),(3, '2013-02-01', 4000),
(1, '2013-01-01', 4500),(2, '2013-01-01', 3500);
select *from incentive;

output:

Result Grid | Filter Rows: | Export:

Employee_ref_id	Incentive_date	Incentive_amount
1	2013-02-01	5000
2	2013-02-01	3000
3	2013-02-01	4000
1	2013-01-01	4500
2	2013-01-01	3500

a) Get First_Name from employee table using Tom name "Employee Name".

Query: select last_Name from employee where first_name = 'Tom';

Output:

Result Grid

last_Name
Jose

b) Get FIRST_NAME, Joining Date, and Salary from employee table.

Query: select First_name,Joining_date,salary from employee;

Output :

Result Grid | Filter Rows: | Export:

First_name	Joining_date	salary
John	2013-01-13 12:00:00	1000000
Michael	2013-01-13 12:00:00	800000
Roy	2013-02-13 12:00:00	700000
Tom	2013-02-13 12:00:00	600000
Jerry	2013-02-13 12:00:00	650000
Philip	2013-01-13 12:00:00	750000
TestNamel	2013-01-13 12:00:00	650000
TestName2	2013-02-13 12:00:00	600000
veer	2013-01-13 12:00:00	1000000

c) Get all employee details from the employee table order by First_Name Ascending and Salary descending?

Query: select*from employee order by First_name asc,salary desc;

Output:

Result Grid | Filter Rows: | Edit: | Export/Import:

Employee_id	First_name	Last_name	Salary	Joining_date	Department
5	Jerry	Pinto	650000	2013-02-13 12:00:00	Insurance
1	John	Abraham	1000000	2013-01-13 12:00:00	Banking
2	Michael	Clarke	800000	2013-01-13 12:00:00	Insurance
6	Philip	Mathew	750000	2013-01-13 12:00:00	Services
3	Roy	Thomas	700000	2013-02-13 12:00:00	Banking
8	TestName2	Lname%	600000	2013-02-13 12:00:00	Insurance
7	TestName1	123	650000	2013-01-13 12:00:00	Services
4	Tom	Jose	600000	2013-02-13 12:00:00	Insurance
9	veer	Abraham	1000000	2013-01-13 12:00:00	Banking
NULL	NULL	NULL	NULL	NULL	NULL

d) Get employee details from employee table whose first name contains 'J'.

Query: select*from employee where First_name like "j%";

Output:

Result Grid | Filter Rows: | Edit: | Export/Import:

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-13 12:00:00	Banking
5	Jerry	Pinto	650000	2013-02-13 12:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

e) Get department wise maximum salary from employee table order by salary ascending?

Query:

```
select Department, max(Salary)
from employee
group by Department
order by Max(Salary);
```

output:

Result Grid | Filter Rows:

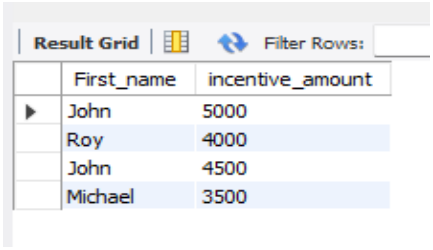
Department	Max_Salary
Services	750000
Insurance	800000
Banking	1000000

f) Select first_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000

Query:

```
select employee.First_name,Incentive.incentive_amount
from Employee join incentive
on employee.Employee_id=incentive.employee_ref_id
where incentive_amount>3000;
```

output:



Result Grid | Filter Rows:

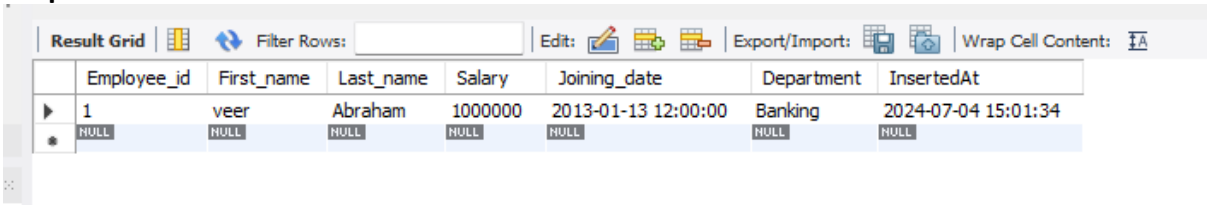
First_name	incentive_amount
John	5000
Roy	4000
John	4500
Michael	3500

g) Create After Insert trigger on Employee table which insert records in view table

Query:

```
create table employeeview (Employee_id INT primary key auto_increment,
 First_name VARCHAR(50),Last_name VARCHAR(50),  Salary INT,
 Joining_date datetime,Department VARCHAR(50), InsertedAt datetime);
delimiter //
create trigger insertview
after insert
on employee
for each row
begin
insert into employeeview(First_name, Last_name, Salary, Joining_date,
Department,InsertedAt)values(new.First_name, new.Last_name,New.
Salary,new.Joining_date,New.Department,now());
end //
delimiter ;
insert into  Employee (First_name, Last_name, Salary, Joining_date, Department)
values
('veer', 'Abraham', 1000000, '2013-01-13 12.00.00', 'Banking');
select*from employeeview;
```

output:



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

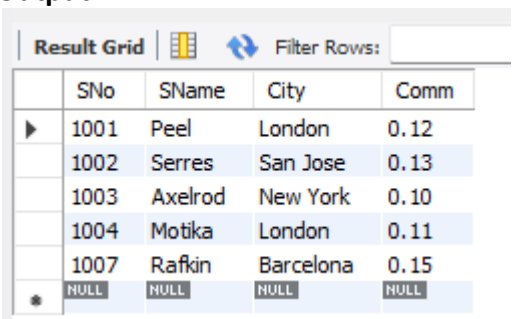
Employee_id	First_name	Last_name	Salary	Joining_date	Department	InsertedAt
1	veer	Abraham	1000000	2013-01-13 12:00:00	Banking	2024-07-04 15:01:34
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Q-4 . Create table given below: Salesperson and Customer

Query:

```
create table Salesperson ( SNo int primary key,SName varchar(50),
City varchar(50),Comm decimal(3, 2));
Insert into Salesperson (SNo, SName, City, Comm) values
(1001, 'Peel', 'London', 0.12),
(1002, 'Serres', 'San Jose', 0.13),
(1004, 'Motika', 'London', 0.11),
(1007, 'Rafkin', 'Barcelona', 0.15),
(1003, 'Axelrod', 'New York', 0.10);
```

Output:

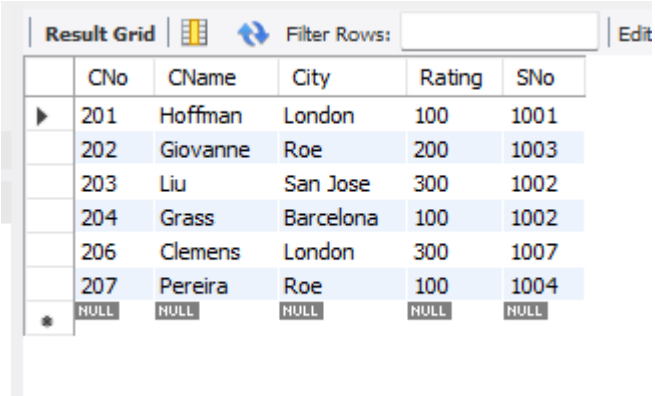


Result Grid | Filter Rows:

SNo	SName	City	Comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1003	Axelrod	New York	0.10
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15
NULL	NULL	NULL	NULL

```
create table Customer (CNo int primary key,CName varchar(50),City varchar(50),
   Rating int,SNo int,foreign key (SNo) references Salesperson(SNo));
Insert into Customer (CNo, CName, City, Rating, SNo) values
(201, 'Hoffman', 'London', 100, 1001),
(202, 'Giovanne', 'Roe', 200, 1003),
(203, 'Liu', 'San Jose', 300, 1002),
(204, 'Grass', 'Barcelona', 100, 1002),
(206, 'Clemens', 'London', 300, 1007),
(207, 'Pereira', 'Roe', 100, 1004);
```

Output:



Result Grid | Filter Rows: | Edit

CNo	CName	City	Rating	SNo
201	Hoffman	London	100	1001
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
204	Grass	Barcelona	100	1002
206	Clemens	London	300	1007
207	Pereira	Roe	100	1004
NULL	NULL	NULL	NULL	NULL

a) Names and cities of all salespeople in London with commission above 0.12

Query: select SName, City from Salesperson where City = 'London' and Comm > 0.12;

Output: No result find

b) All salespeople either in Barcelona or in London

Query : select *from Salesperson where city = 'Barcelona' or city = 'London';

Output:

Result Grid | Filter Rows:

SNo	SName	City	Comm
1001	Peel	London	0.12
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15
NULL	NULL	NULL	NULL

c) All salespeople with commission between 0.10 and 0.12.

Query: select*from salesperson where comm between 0.10 and 0.12;

Output:

Result Grid | Filter Rows:

SNo	SName	City	Comm
1001	Peel	London	0.12
1003	Axelrod	New York	0.10
1004	Motika	London	0.11
NULL	NULL	NULL	NULL

d) All customers excluding those with rating <= 100 unless they are located in Rome

Query: select *from customer where rating > 100 or city = 'Rome';

Output:

Result Grid | Filter Rows:

CNo	CName	City	Rating	SNo
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
206	Clemens	London	300	1007
NULL	NULL	NULL	NULL	NULL