

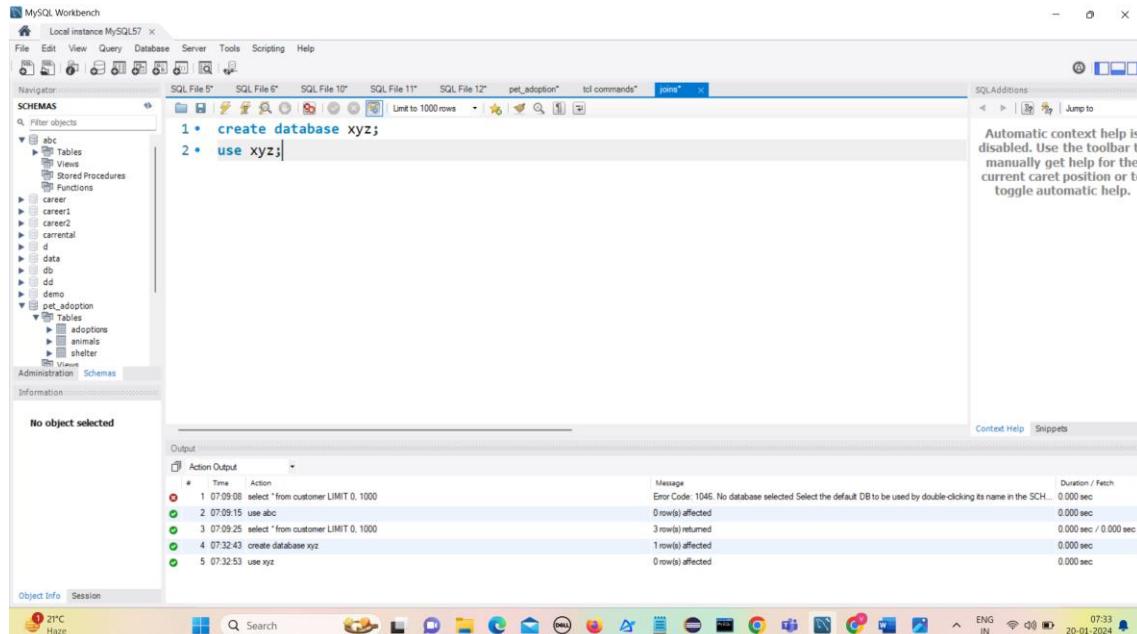
Assignment-4

Joins

Join statement is used to combine data or rows from two or more tables based on a common field between them.

1) INNER JOIN

Create a table xyz and use the database to make it as the current working database.



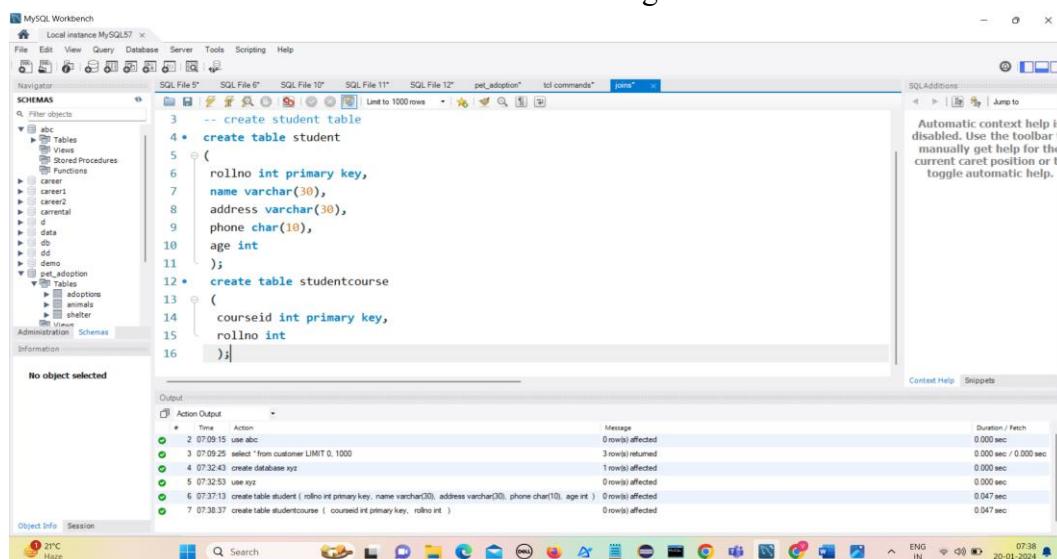
The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'SCHEMAS', there is a tree view with several databases like abc, career, career1, career2, career3, d, data, db, dd, demo, and pet_adoption. Under 'pet_adoption', there are tables for adoptions, animals, and shelter. The 'Information' section at the bottom shows 'No object selected'. In the main query editor window, the following SQL commands are entered:

```
1 • create database xyz;
2 • use xyz;
```

The output pane shows the execution results:

Action	Time	Message	Duration / Fetch
1 07:09:08 select * from customer LIMIT 0, 1000		Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCH...	0.000 sec
2 07:09:15 use abc		0 row(s) affected	0.000 sec
3 07:09:25 select * from customer LIMIT 0, 1000		3 row(s) returned	0.000 sec / 0.000 sec
4 07:32:43 create database xyz		1 row(s) affected	0.000 sec
5 07:32:53 use xyz		0 row(s) affected	0.000 sec

Create two tables Student and Studentcourse using create table statement.



The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'SCHEMAS', there is a tree view with several databases like abc, career, career1, career2, career3, d, data, db, dd, demo, and pet_adoption. Under 'pet_adoption', there are tables for adoptions, animals, and shelter. The 'Information' section at the bottom shows 'No object selected'. In the main query editor window, the following SQL commands are entered:

```
3 -- create student table
4 • create table student
5 (
6   rollno int primary key,
7   name varchar(30),
8   address varchar(30),
9   phone char(10),
10  age int
11 );
12 • create table studentcourse
13 (
14   courseid int primary key,
15   rollno int
16 );
```

The output pane shows the execution results:

Action	Time	Message	Duration / Fetch
2 07:09:15 use abc		0 row(s) affected	0.000 sec
3 07:09:25 select * from customer LIMIT 0, 1000		3 row(s) returned	0.000 sec / 0.000 sec
4 07:32:43 create database xyz		1 row(s) affected	0.000 sec
5 07:32:53 use xyz		0 row(s) affected	0.000 sec
6 07:37:13 create table student (rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int)		0 row(s) affected	0.047 sec
7 07:38:37 create table studentcourse (courseid int primary key, rollno int)		0 row(s) affected	0.047 sec

Insert records into Student and Studentcourse table using insert statement.

```

MySQL Workbench - Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
Tables Views Stored Procedures Functions
career career1 career2 currentel d data db dd demo pet_adoption
Tables adoption animals shelter
Administration Schemas
Information No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* pane* 
15
16
17 -- insert records into student table
18 * insert into student values(1,'ganga','Bodhan','7878787878',22),
19 (2,'Vinita','Nizamabad','1212121212',26),
20 (3,'Lekha','Hyderabad','6565656565',24),
21 (4,'Ram','pune','9898989898',19),
22 (5,'Mona','Mumbai','2121212121',10);
-- insert records into studentcourse table
23 insert into studentcourse values(101,1),
24 (102,3),
25 (103,4),
26 (104,2),
27 (105,1);

Output Action Output
1 Action
4 07-32-43 create database xyz
5 07-32-53 use xyz
6 07-37-13 create table student ( rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int )
7 07-38-37 create table studentcourse ( coursed int primary key, rollno int )
8 07-42-37 insert into student values(1,'ganga','Bodhan','7878787878',22), (2,'Vinita','Nizamabad','1212121212',26), (3,'Lekha','Hyderabad','6565656565',24), (4,'Ram','pune','9898989898',19), (5,'Mona','Mumbai','2121212121',10)
9 07-43-45 insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)

Message
1 rows(s) affected
0 rows(s) affected
0 rows(s) affected
0 rows(s) affected
0 rows(s) affected
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
Duration / Fetch
0.000 sec
0.000 sec
0.047 sec
0.047 sec
0.047 sec
0.000 sec
0.000 sec
0.000 sec
0.000 sec
0.000 sec
Context Help Snippets
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object Info Session
21°C Haze Search ENG IN 20-01-2024 07:44

```

Retrieve records from student and studentcourse table

```

MySQL Workbench - Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
Tables Views Stored Procedures Functions
career career1 career2 currentel d data db dd demo pet_adoption
Tables adoption animals shelter
Administration Schemas
Information No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* pane* 
22 (5,'Mona','Mumbai','2121212121',10);
-- insert records into studentcourse table
23 insert into studentcourse values(101,1),
24 (102,3),
25 (103,4),
26 (104,2),
27 (105,1);
28
29 * select * from student;

Result Grid
rollno name address phone age
1 Ganga Bodhan 7878787878 22
2 Vinita Nizamabad 1212121212 26
3 Lekha Hyderabad 6565656565 24
4 Ram pune 9898989898 19
5 Mona Mumbai 2121212121 10
student 1

Output Action Output
1 Action
5 07-32-53 use xyz
6 07-37-13 create table student ( rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int )
7 07-38-37 create table studentcourse ( coursed int primary key, rollno int )
8 07-42-37 insert into student values(1,'ganga','Bodhan','7878787878',22), (2,'Vinita','Nizamabad','1212121212',26), (3,'Lekha','Hyderabad','6565656565',24), (4,'Ram','pune','9898989898',19), (5,'Mona','Mumbai','2121212121',10)
9 07-43-45 insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)
10 07-45-14 select * from student LIMIT 0, 1000

Message
0 rows(s) affected
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
Duration / Fetch
0.000 sec
0.000 sec
0.047 sec
0.047 sec
0.047 sec
0.000 sec
0.000 sec
0.000 sec
0.000 sec
0.000 sec
Context Help Snippets
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object Info Session
Tomorrow's high Near record Search ENG IN 20-01-2024 07:45

```

```

MySQL Workbench - Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
Tables Views Stored Procedures Functions
career career1 career2 currentel d data db dd demo pet_adoption
Tables adoption animals shelter
Administration Schemas
Information No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* pane* 
23 -- insert records into studentcourse table
24 * insert into studentcourse values(101,1),
25 (102,3),
26 (103,4),
27 (104,2),
28 (105,1);
29 * select * from student;
30 * select * from studentcourse;

Result Grid
coursed rollno
101 1
102 3
103 4
104 2
105 1
studentcourse 2

Output Action Output
1 Action
6 07-37-13 create table student ( rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int )
7 07-38-37 create table studentcourse ( coursed int primary key, rollno int )
8 07-42-37 insert into student values(1,'ganga','Bodhan','7878787878',22), (2,'Vinita','Nizamabad','1212121212',26), (3,'Lekha','Hyderabad','6565656565',24), (4,'Ram','pune','9898989898',19), (5,'Mona','Mumbai','2121212121',10)
9 07-43-45 insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)
10 07-45-14 select * from student LIMIT 0, 1000
11 07-45-47 select * from studentcourse LIMIT 0, 1000

Message
0 rows(s) affected
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0
Duration / Fetch
0.047 sec
0.047 sec
0.047 sec
0.000 sec
Context Help Snippets
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object Info Session
Tomorrow's high Near record Search ENG IN 20-01-2024 07:45

```

Inner join

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:30 • select * from studentcourse;
31
32
33
34
35 -- Names and age of students enrolled in different courses
36 • select s.name , s.age from student s inner join studentcourse c
on s.rollno=c.rollno;The results grid shows the following data:| name | age |
| --- | --- |
| ganga | 22 |
| Latha | 24 |
| Ram | 19 |
| Virth | 26 |
| ganga | 22 |

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
create table studentcourse (courseid int primary key, rollno int)	7 07:38:37	0 rows affected	0.047 sec
insert into student values('1','ganga','Bodhan','7878787878','22')	8 07:42:45	5 rows affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
insert into studentcourse values('101.1'), ('102.3'), ('103.4'), ('104.2'), ('105.1')	9 07:43:45	5 rows affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
select * from student LIMIT 0, 1000	10 07:45:14	5 rows returned	0.000 sec / 0.000 sec
select * from studentcourse LIMIT 0, 1000	11 07:45:47	5 rows returned	0.000 sec / 0.000 sec
select s.name , s.age from student s inner join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	12 07:50:14	5 rows returned	0.000 sec / 0.000 sec

This query gives the names and age of students enrolled in different courses.

2) LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*.

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:37 on s.rollno=c.rollno;
38
39
40
41 -- LEFT JOIN
42 • select s.name , s.age from student s
left join studentcourse c
on s.rollno=c.rollno;The results grid shows the following data:| name | age |
| --- | --- |
| ganga | 22 |
| Latha | 24 |
| Ram | 19 |
| Virth | 26 |
| ganga | 22 |
| Mona | 10 |

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
insert into student values('1','ganga','Bodhan','7878787878','22')	8 07:42:37	5 rows affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
insert into studentcourse values('101.1'), ('102.3'), ('103.4'), ('104.2'), ('105.1')	9 07:43:45	5 rows affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
select * from student LIMIT 0, 1000	10 07:45:14	5 rows returned	0.000 sec / 0.000 sec
select * from studentcourse LIMIT 0, 1000	11 07:45:47	5 rows returned	0.000 sec / 0.000 sec
select s.name , s.age from student s inner join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	12 07:50:14	5 rows returned	0.000 sec / 0.000 sec
select s.name , s.age from student s left join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	13 08:24:43	6 rows returned	0.000 sec / 0.000 sec

3) RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.

For the rows for which there is no matching row on the left side, the result-set will contain *null*

```
42 •    select s.name , s.age from student s
43     left join studentcourse c
44     on s.rollno=c.rollno;
45     -- RIGHT JOIN
46 •    select s.name , s.age from student s
47     right join studentcourse c
48     on s.rollno=c.rollno;
```

name	age
ganga	22
Leha	24
Ram	19
Vinit	26
ganga	22

Action Output:

#	Time	Action	Message	Duration / Fetch
9	07:43:45	insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
10	07:45:14	select * from student LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
11	07:45:14	select * from studentcourse LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
12	07:50:14	select s.name , s.age from student s inner join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
13	08:24:43	select s.name , s.age from student s left join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
14	08:26:41	select s.name , s.age from student s right join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

4) FULL JOIN

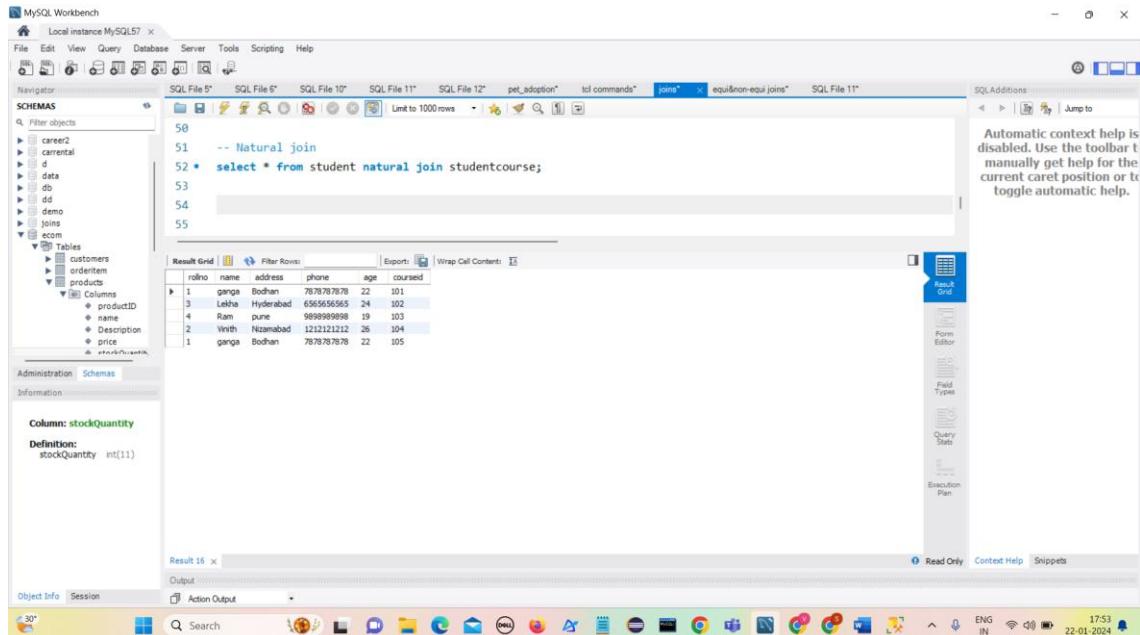
FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

```
51
52
53     -- Full join
54 •    select * from student
55     full join studentcourse ;
```

rollno	name	address	phone	age	gender	rollno
1	ganga	Bodhan	7879787979	22	101	3
2	Vinit	Nizamabad	1212121212	26	101	1
3	Leha	Hyderabad	6565655656	24	101	1
4	Ram	pune	9899999999	19	101	1
5	Mona	Mumbai	2121212121	10	101	1
1	ganga	Bodhan	7879787979	22	102	3
2	Vinit	Nizamabad	1212121212	26	102	3
3	Leha	Hyderabad	6565655656	24	103	4
4	Ram	pune	9899999999	19	103	4
5	Mona	Mumbai	2121212121	10	103	4
1	ganga	Bodhan	7879787979	22	104	2
2	Vinit	Nizamabad	1212121212	26	104	2
3	Leha	Hyderabad	6565655656	24	104	2
4	Ram	pune	9899999999	19	104	2
5	Mona	Mumbai	2121212121	10	104	2
1	ganga	Bodhan	7879787979	22	105	1

5) NATURAL JOIN

A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.



The screenshot shows the MySQL Workbench interface with a query editor window. The SQL code is:

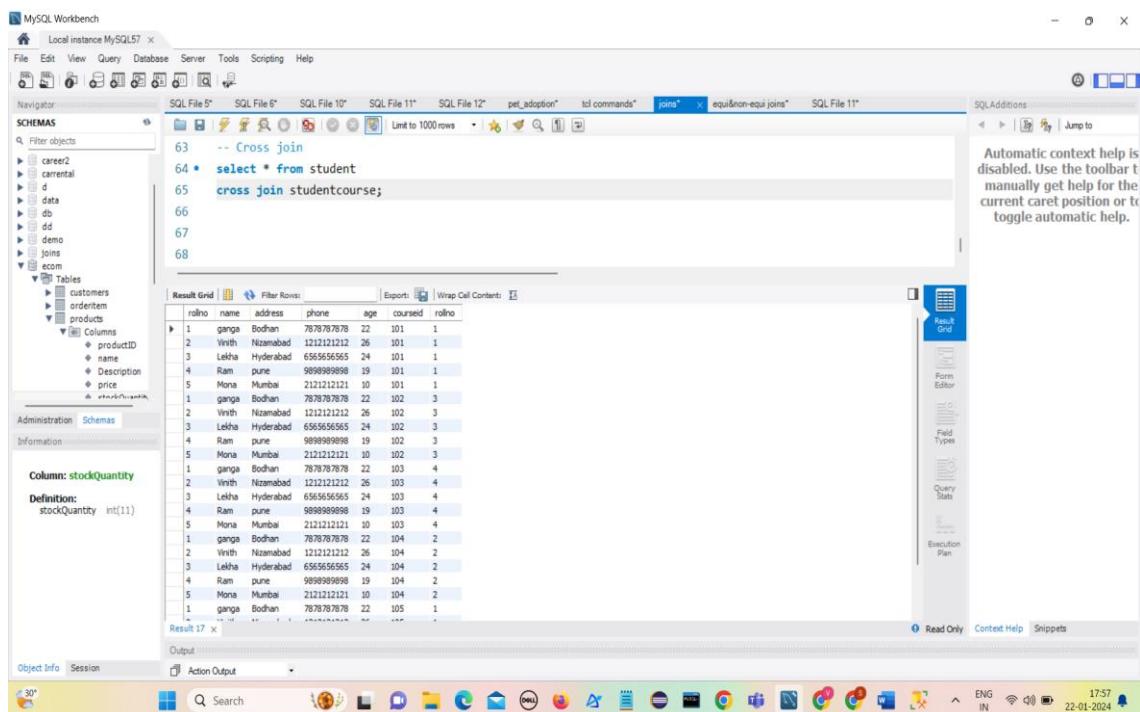
```
50
51 -- Natural join
52 • select * from student natural join studentcourse;
53
54
55
```

The result grid shows the following data:

rollno	name	address	phone	age	coursed
1	ganga	Bodhan	7878787878	22	101
2	Vinith	Nizamabad	1212121212	26	101
3	Leeha	Hyderabad	6565656565	24	102
4	Ram	pune	9898989898	19	103
2	Vinith	Nizamabad	1212121212	26	104
1	ganga	Bodhan	7878787878	22	105

6) CROSS JOIN

A *cross join* is a type of join that returns the Cartesian product of rows from the tables in the join. It combines each row from the first table with each row from the second table.



The screenshot shows the MySQL Workbench interface with a query editor window. The SQL code is:

```
63 -- Cross join
64 • select * from student
65 cross join studentcourse;
66
67
68
```

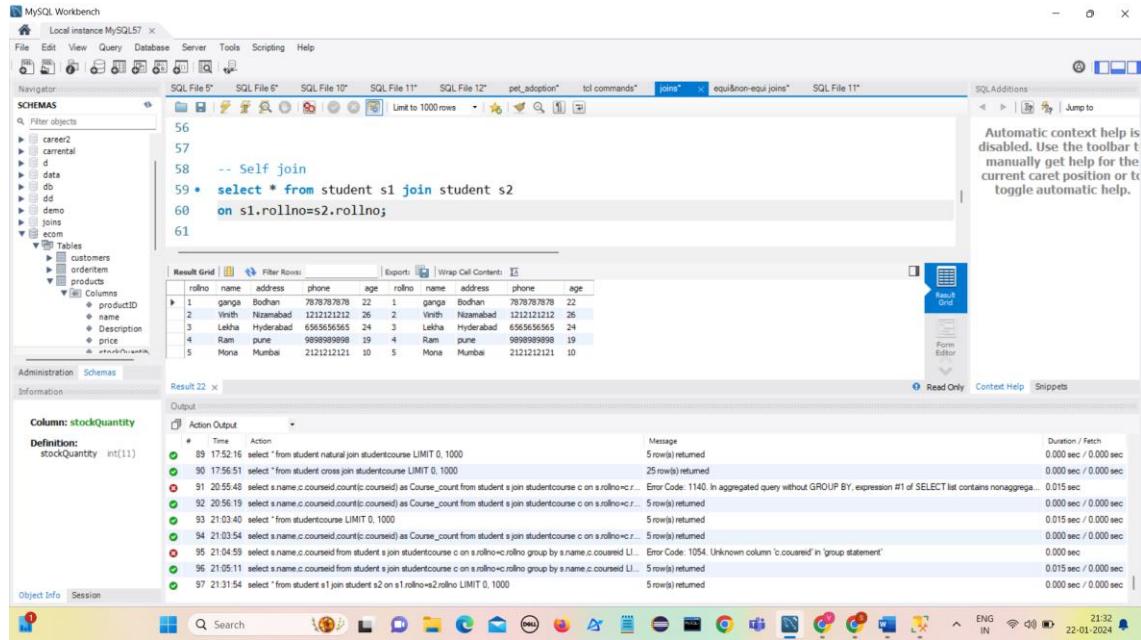
The result grid shows the following data, illustrating the Cartesian product:

rollno	name	address	phone	age	coursed	rollno
1	ganga	Bodhan	7878787878	22	101	1
2	Vinith	Nizamabad	1212121212	26	101	1
3	Leeha	Hyderabad	6565656565	24	101	1
4	Ram	pune	9898989898	19	101	1
5	Mona	Mumbai	2121212121	10	101	1
1	ganga	Bodhan	7878787878	22	102	3
2	Vinith	Nizamabad	1212121212	26	102	3
3	Leeha	Hyderabad	6565656565	24	102	3
4	Ram	pune	9898989898	19	102	3
5	Mona	Mumbai	2121212121	10	102	3
1	ganga	Bodhan	7878787878	22	103	4
2	Vinith	Nizamabad	1212121212	26	103	4
3	Leeha	Hyderabad	6565656565	24	103	4
4	Ram	pune	9898989898	19	103	4
5	Mona	Mumbai	2121212121	10	103	4
1	ganga	Bodhan	7878787878	22	104	2
2	Vinith	Nizamabad	1212121212	26	104	2
3	Leeha	Hyderabad	6565656565	24	104	2
4	Ram	pune	9898989898	19	104	2
5	Mona	Mumbai	2121212121	10	104	2
1	ganga	Bodhan	7878787878	22	105	1

7) SELF JOIN

A self join is a regular join that is used to join a table with itself.

It basically allows us to combine the rows from the same table based on some specific conditions.



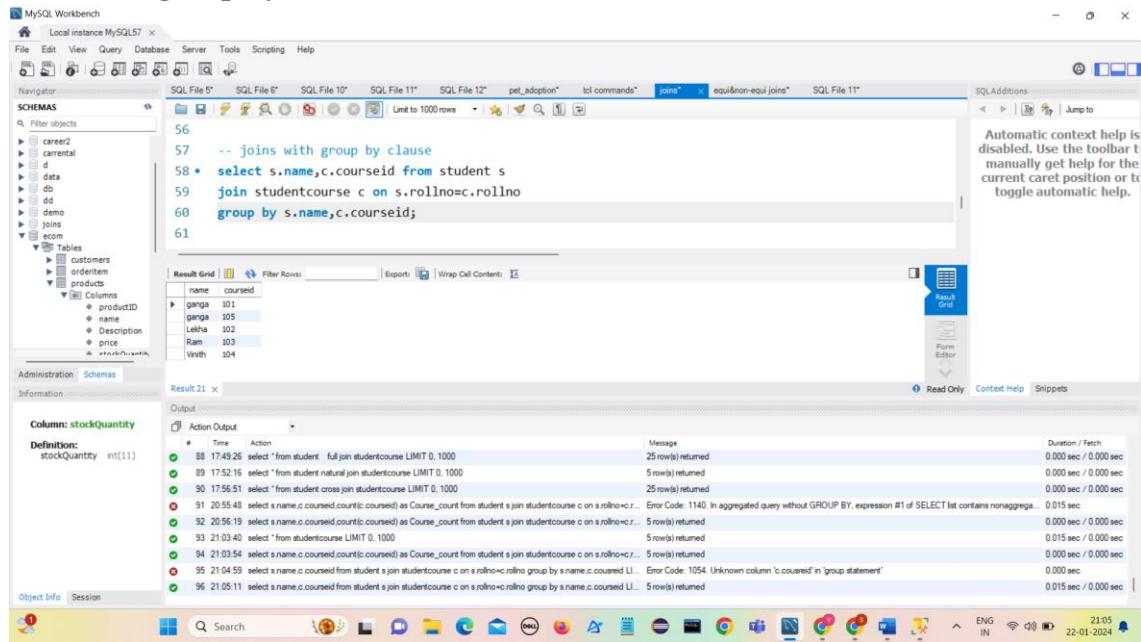
The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with tables like career2, currental, d, data, dd, demo, joins, and ecom.
- SQL Editor:** Contains the following SQL code:


```

56
57
58 -- Self join
59 • select * from student s1 join student s2
60 on s1.rollno=s2.rollno;
61
      
```
- Result Grid:** Displays the results of the self join query, showing five rows of student data.
- Output:** Shows the execution history with 97 entries, mostly related to the self join query.

8) Joins with group by clause



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with tables like career2, currental, d, data, dd, demo, joins, and ecom.
- SQL Editor:** Contains the following SQL code:


```

56
57 -- joins with group by clause
58 • select s.name,c.courseid from student s
59 join studentcourse c on s.rollno=c.rollno
60 group by s.name,c.courseid;
61
      
```
- Result Grid:** Displays the results of the query, showing five rows of student-course combinations.
- Output:** Shows the execution history with 96 entries, mostly related to the join with group by clause query.

9) Join with aggregate functions

Aggregate functions like SUM, AVG , COUNT, MIN,MAX can be used with joins.

Retrieve the count of courses each student is enrolled in

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query:

```
56 -- Retrieve the count of courses each student is enrolled in.
57 • select s.name,c.courseid,count(c.courseid) as Course_count
58   from student s join studentcourse c
59   on s.rollno=c.rollno
60   group by s.name,c.courseid;
61
```
- Result Grid:** Shows the output of the query:

name	courseid	Course_count
panga	101	1
panga	105	1
Leisha	102	1
Ram	103	1
Vnith	104	1
- Action Output:** Shows the history of actions taken in the session, including various SELECT and UPDATE statements.
- System Bar:** Includes the Windows taskbar with icons for various applications like File Explorer, Task View, and Start.

10) Joins with group by and having clause

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query:

```
75 -- Joins with group by and having clause
76 -- Retrieve names of students enrolled in more than 1 course
77 • select s.name,count(c.courseid) as Course_count from student s
78 join studentcourse c on s.rollno=c.rollno
79 group by s.name
80 having Course_count>1;
81
```
- Result Grid:** Shows the output of the query:

name	Course_count
panga	2
- Action Output:** Shows the history of actions taken in the session, including various SELECT and UPDATE statements.
- System Bar:** Includes the Windows taskbar with icons for various applications like File Explorer, Task View, and Start.

Equi joins & Non-Equi Joins

Create database:

create database joins;

Use database:

use joins; make joins as the current working database

The screenshot shows the MySQL Workbench interface. In the 'Schemas' tree on the left, a new database named 'joins' is visible under 'abc'. The 'Session' tab at the bottom indicates that 'joins' is the current database. The 'Output' pane displays the SQL commands run:

```
1 11:09:24 select * from student full join studentcourse on student.rollno=studentcourse.rollno LIMIT 0, 1000
2 12 09:27:23 SELECT * FROM student FULL JOIN studentcourse ON student.rollno = studentcourse.rollno LIMIT 0, 1000
3 13 09:27:23 SELECT * FROM student FULL JOIN studentcourse LIMIT 0, 1000
4 14 09:28:14 select * from student full join studentcourse LIMIT 0, 1000
5 15 09:30:35 select * from student left join studentcourse on student.rollno=studentcourse.rollno union select * from student...
6 16 09:31:35 select * from student full join studentcourse on student.rollno=studentcourse.rollno LIMIT 0, 1000
7 17 09:32:04 select * from student left join studentcourse – on student.rollno=studentcourse.rollno union select * from student...
8 18 10:22:30 select * from student full join studentcourse LIMIT 0, 1000
9 19 10:30:44 create database joins
10 20 10:30:55 use joins
```

Create table Student

Create table Record

The screenshot shows the MySQL Workbench interface. The 'Schemas' tree on the left shows the 'joins' database selected. The 'Session' tab at the bottom indicates that 'joins' is the current database. The 'Output' pane displays the SQL commands run:

```
3 -- Create table Student
4 4 * create table Student
5 5 *
6 6 id int primary key,
7 7 name varchar(30),
8 8 class int,
9 9 city varchar(30)
10 10 );
11 -- Create table Record
12 12 * create table Record
13 13 *
14 14 id int,
15 15 class varchar(30),
16 16 city varchar(30)
17 17 )
18 18
```

Below the command history, the 'Output' pane shows the results of the queries run:

```
16 10:39:15 select * from student full join studentcourse on student.rollno=studentcourse.rollno LIMIT 0, 1000
17 10:39:15 Error Code: 1054. Unknown column 'student.rollno' in 'on clause'
18 10:39:15 Error Code: 1054. Unknown column 'student.rollno' in 'on clause'
19 10:39:24 select * from student left join studentcourse – on student.rollno=studentcourse.rollno union select * from student...
20 10:39:24 25 row(s) returned
21 10:39:24 1 row(s) affected
22 10:39:24 0 row(s) affected
23 10:42:51 0 row(s) affected
24 10:46:56 create table Student ( id int primary key, name varchar(30), class int, city varchar(30) )
25 10:46:57 create table Record ( id int, class varchar(30), city varchar(30) )
```

Insert records into Student and Record Table

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema 'studentcourse' with various objects like tables (student, record), views, and stored procedures. The main area shows a query editor with the following SQL code:

```
SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* t1 commands* joins* regular+equal joins*
26
27 -- Insert records into Student table
28 • insert into Student values(3,'Hina',3,'Delhi');
29 (4,'Megha',2,'Delhi');
30 (5,'Gouri',2,'Delhi');
31
32 -- Insert records into Record table
33 • insert into Record values(9,3,'Delhi');
34 (10,2,'Delhi');
35 (11,2,'Delhi');
36
37
38
39
40
41
```

A context help window titled 'Automatic context help is disabled.' is overlaid on the right side of the screen, with the message 'Use the toolbar manually get help for the current caret position or toggle automatic help.'

Retrieve records from Student table using select statement

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables like Record, Student, and OrderLine. The main area contains an SQL editor with the following code:

```

22
23 -- Insert records into Record table
24 • insert into Record values(9,3,'Delhi'),
25 (10,2,'Delhi'),
26 (12,2,'Delhi')
27
28 • select * from Student;
29

```

The results grid shows the following data for the Student table:

ID	Name	Class	City
1	Amit	3	Delhi
2	Megha	2	Delhi
3	Gauri	2	Delhi

The bottom section shows a timeline of recent actions:

- 79 15:12:26 select * from ordertemp LIMIT 0, 1000
- 80 15:19:41 select * from ordertemp LIMIT 0, 1000
- 81 15:34:10 update products set name='Bell' where productid=4 select produced from ordertemp
- 82 15:34:10 update products set name='Bell' where productid=4 select produced from ordertemp where itemno: 1 rows(s) affected 0 rows(s) matched: 1 Changed: 1 Warnings: 0
- 83 15:34:43 select * from products LIMIT 0, 1000
- 84 17:31:47 use joins
- 85 17:32:21 select * from Student LIMIT 0, 1000

Retrieve records from Record table using select statement

The screenshot shows the MySQL Workbench application window. At the top, there's a menu bar with File, Edit, View, Query, Database, Server, Tools, Scripting, Help. Below the menu is a toolbar with various icons for database management. The main area has tabs for SQL File 5*, SQL File 6*, SQL File 10*, SQL File 11*, SQL File 12*, pet_adoption*, tcl commands*, joins*, and equalJoin-equal joins*. The SQL tab contains the following code:

```
36
37 • select * from Student;
38
39 • select * from Record;
40
41
42
43
44
```

Below the code is a Result Grid table with columns id, Class, and City. The data is as follows:

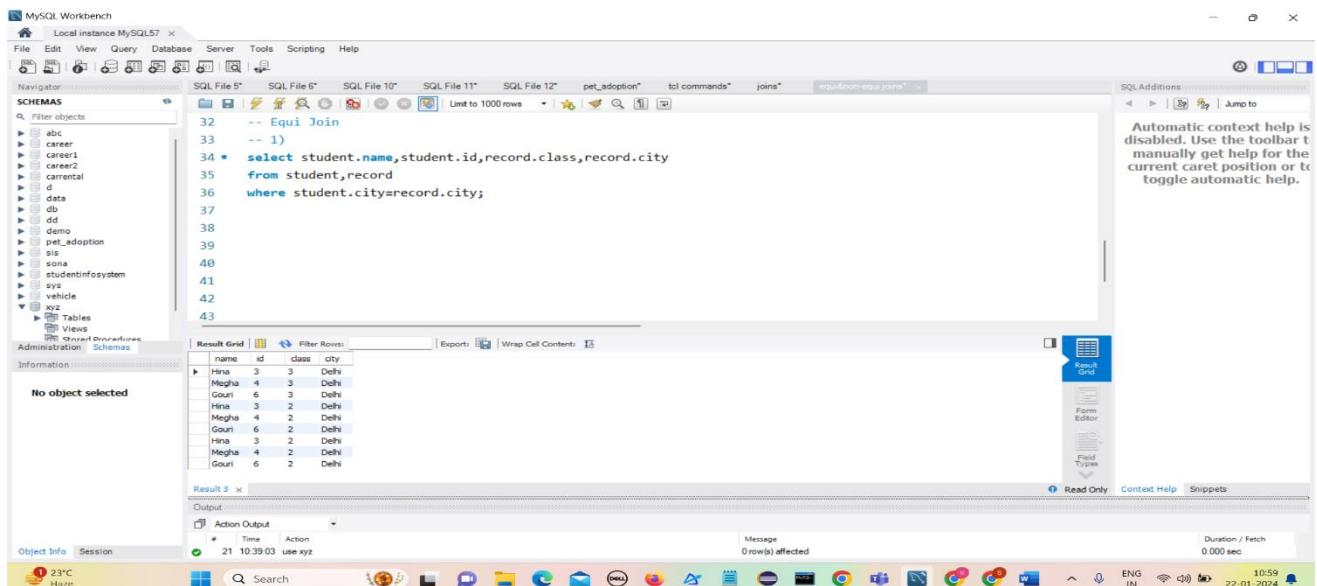
	Class	City
9	2	Delhi
10	2	Delhi
12	2	Delhi

At the bottom left, there's an Information panel with 'No object selected'. The bottom right shows a status bar with '10:55', 'ENG IN_4', '22-01-2024', and other system icons.

11) EQUI Join:

EQUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables.

EQUI JOIN also create JOIN by using JOIN with ON and then providing the names of the columns with their relative tables to check equality using equal sign (=).



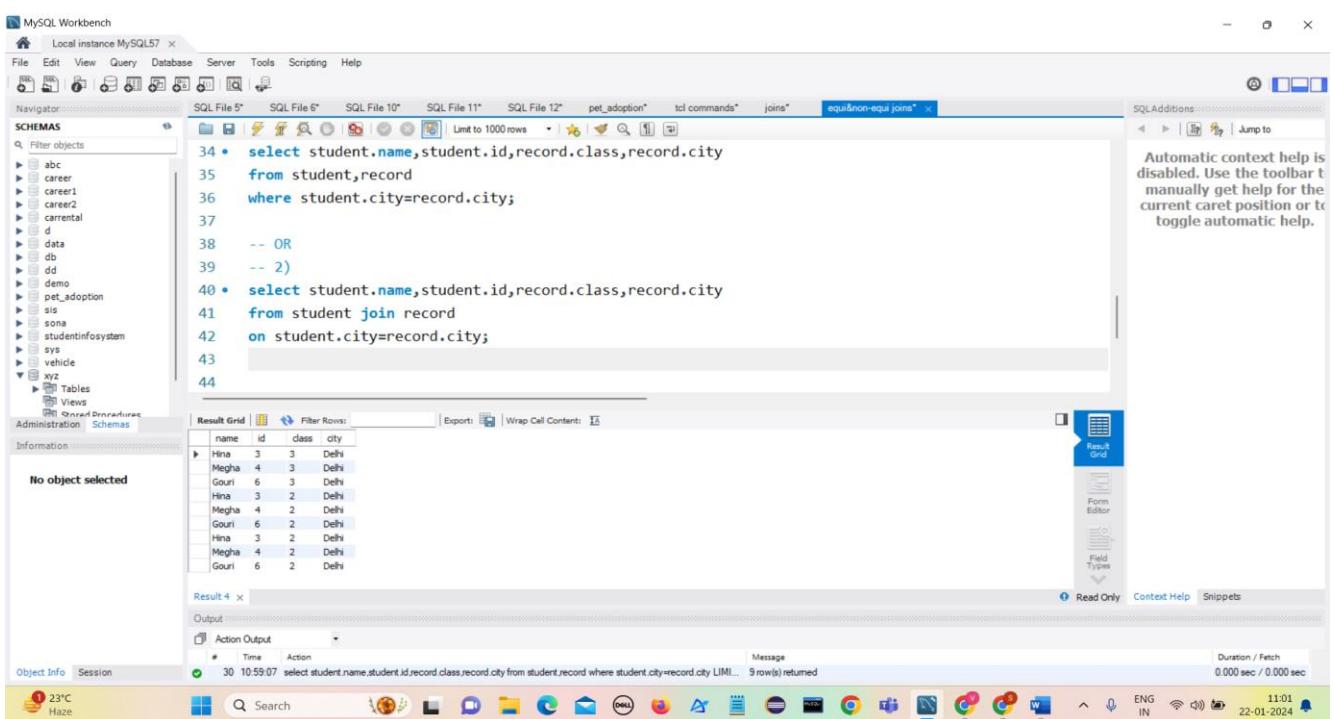
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL57
- Query Editor:** SQL File 5* (containing the EQUI JOIN code)
- Result Grid:** Displays the query results in a tabular format with columns: name, id, class, city. The data shows multiple rows for each student ID, indicating the join operation.
- Message:** 0 row(s) affected
- Action Output:** Shows the time (21:10:39.03) and action (use xyz).
- System Bar:** Shows the date (22-01-2024), time (10:59), and system status (23°C Haze).

```
-- Equi Join
-- 1)
34 • select student.name,student.id,record.class,record.city
from student,record
where student.city=record.city;
```

	name	id	class	city
1	Hina	3	3	Delhi
2	Megha	4	3	Delhi
3	Gouri	6	3	Delhi
4	Hina	3	2	Delhi
5	Megha	4	2	Delhi
6	Gouri	6	2	Delhi
7	Hina	3	2	Delhi
8	Megha	4	2	Delhi
9	Gouri	6	2	Delhi

OR



The screenshot shows the MySQL Workbench interface with the following details:

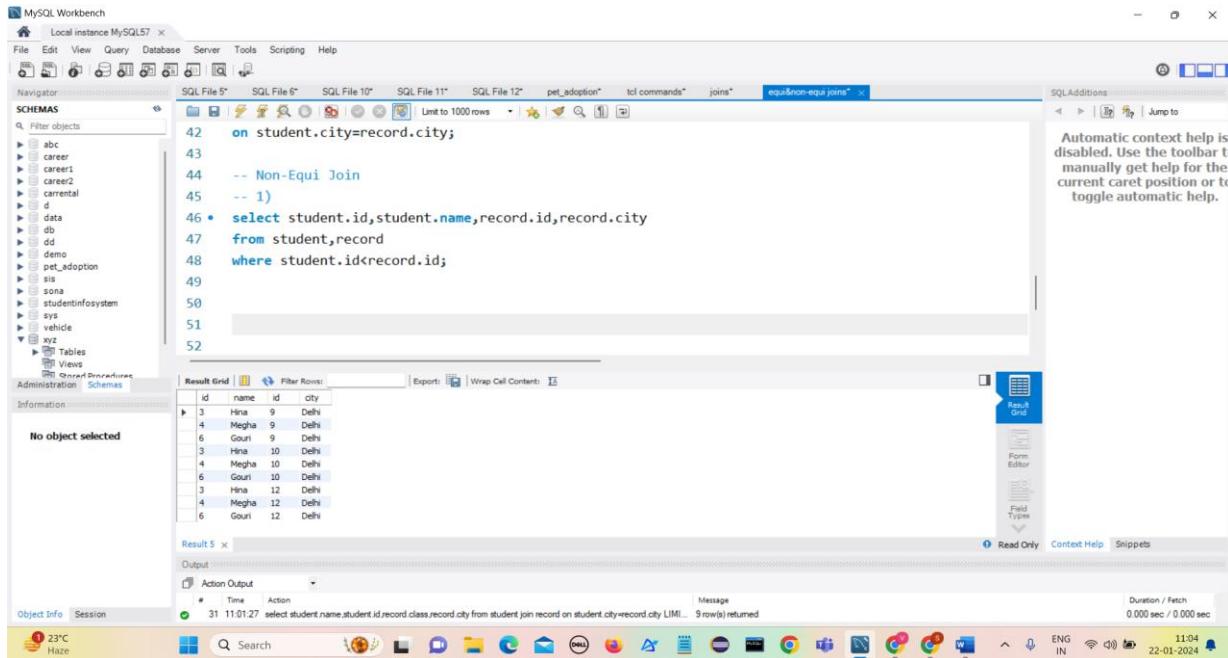
- Schemas:** Local instance MySQL57
- Query Editor:** SQL File 5* (containing the EQUI JOIN code using the ON clause)
- Result Grid:** Displays the query results in a tabular format with columns: name, id, class, city. The data shows multiple rows for each student ID, indicating the join operation.
- Message:** 9 row(s) returned
- Action Output:** Shows the time (30:10:59.07) and action (select student.name,student.id,record.class,record.city from student,record where student.city=record.city LIMIT 9).
- System Bar:** Shows the date (22-01-2024), time (11:01), and system status (23°C Haze).

```
-- OR
-- 2)
40 • select student.name,student.id,record.class,record.city
from student join record
on student.city=record.city;
```

	name	id	class	city
1	Hina	3	3	Delhi
2	Megha	4	3	Delhi
3	Gouri	6	3	Delhi
4	Hina	3	2	Delhi
5	Megha	4	2	Delhi
6	Gouri	6	2	Delhi
7	Hina	3	2	Delhi
8	Megha	4	2	Delhi
9	Gouri	6	2	Delhi

12) Non- EQUI join

NON EQUI JOIN performs a JOIN using comparison operator other than equal(=) sign like >, <, >=, <= with conditions.



The screenshot shows the MySQL Workbench interface with a query editor window titled "equi&non-equi joins*". The code is:

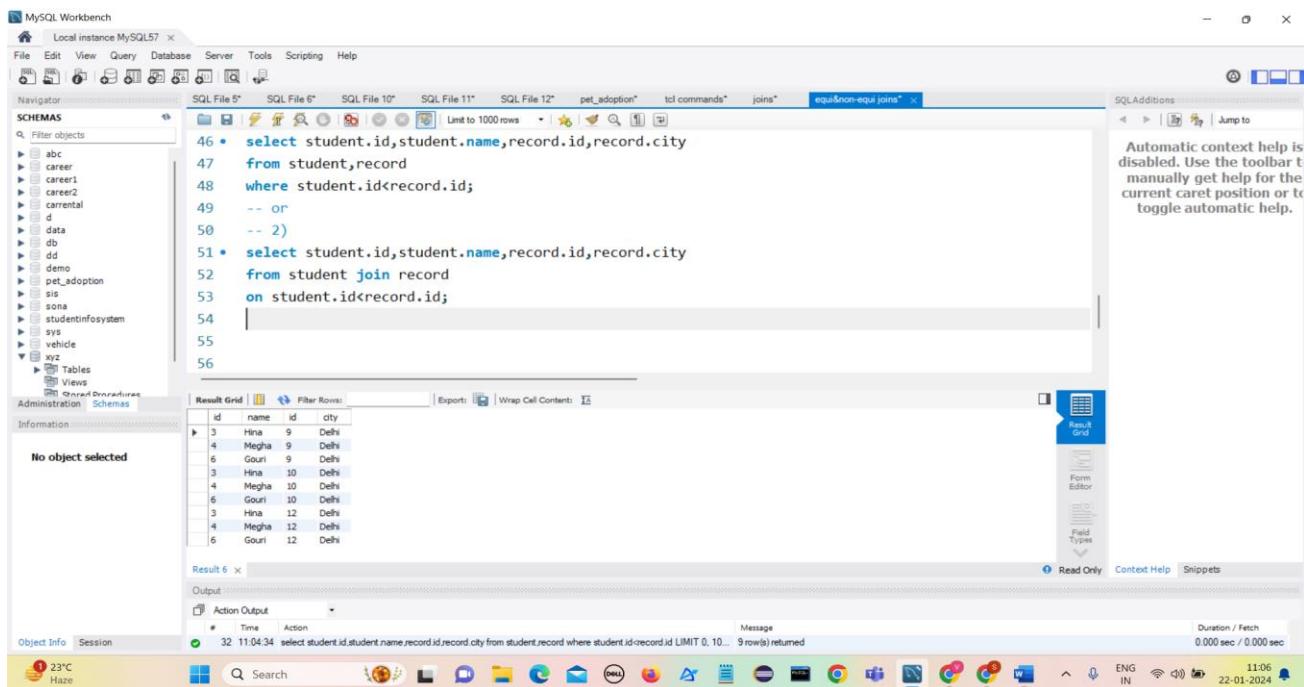
```
42 on student.city=record.city;
43
44 -- Non-EQUI Join
45 -- 1)
46 • select student.id,student.name,record.id,record.city
  from student,record
 where student.id<record.id;
47
48
49
50
51
52
```

The result grid shows the following data:

ID	Name	ID	CITY
3	Hina	9	Delhi
4	Megha	9	Delhi
6	Gouri	9	Delhi
3	Hina	10	Delhi
4	Megha	10	Delhi
6	Gouri	10	Delhi
3	Hina	12	Delhi
4	Megha	12	Delhi
6	Gouri	12	Delhi

The status bar at the bottom indicates the session details: 23°C Haze, 22-01-2024, 11:04, ENG IN.

Or



The screenshot shows the MySQL Workbench interface with a query editor window titled "equi&non-equi joins*". The code is:

```
46 • select student.id,student.name,record.id,record.city
  from student,record
 where student.id<record.id;
47
48 -- or
49 -- 2)
50
51 • select student.id,student.name,record.id,record.city
  from student join record
  on student.id<record.id;
52
53
54
55
56
```

The result grid shows the same data as the previous screenshot:

ID	Name	ID	CITY
3	Hina	9	Delhi
4	Megha	9	Delhi
6	Gouri	9	Delhi
3	Hina	10	Delhi
4	Megha	10	Delhi
6	Gouri	10	Delhi
3	Hina	12	Delhi
4	Megha	12	Delhi
6	Gouri	12	Delhi

The status bar at the bottom indicates the session details: 23°C Haze, 22-01-2024, 11:06, ENG IN.

13) Querying data using Subqueries

Retrieve the products ordered by customers

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: Local instance MySQL57
SCHEMAS: Filter objects
Tables: customers, orderitem, products
Columns: productID, name, Description, price
Result Grid: Filter Rows: Export/Import: Wrap Cell Content: Result Grid
116 • select * from products;
117
118 -- Retrieve the products ordered by customers
119 • select p.* from products p
120 where p.productid= ANY ( select o.productid from orderitem o);
121

```

productID	name	Description	price	stockQuantity
2	Smartphone	Latest smartphone	600	15
3	Tablet	Portable tablet	300	20
4	Headphones	Noise-cancelling	150	30
6	Coffee Maker	Automatic coffee maker	50	25
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10

Output:

Action	Time	Message	Duration / Fetch
97	21:31:54	select * from student s2 join student s on s1.rollno=s2.rollno LIMIT 0, 1000	0.000 sec / 0.000 sec
98	21:38:54	selected s.name, count(c.courseid) as Course_count from student s join studentcourse c on s.rollno=c.rollno group by s.rollno	0.000 sec / 0.000 sec
99	21:39:10	selected s.name, count(c.courseid) as Course_count from student s join studentcourse c on s.rollno=c.rollno group by s.rollno	0.000 sec / 0.000 sec
100	21:53:43	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	Error Code: 1146. Table 'xyz.products' doesn't exist
101	21:53:59	use ecom	0.000 sec
102	21:54:07	selected p.* from products p where p.productid=(select o.productid from orderitem) LIMIT 0, 1000	Error Code: 1054. Unknown column 'o.productid' in field list
103	21:54:22	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	Error Code: 1242. Subquery returns more than 1 row
104	21:54:24	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	Error Code: 1242. Subquery returns more than 1 row
105	21:54:55	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	7 row(s) returned
106	21:57:19	select productid, name from products where productid=(select productid from orderitem where quantity=2)	0.000 sec / 0.000 sec

Retrieve the productid,names of products whose quantity is 2

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: Local instance MySQL57
SCHEMAS: Filter objects
Tables: customers, orderitem, products
Columns: productID, name, Description, price
Result Grid: Filter Rows: Export/Import: Wrap Cell Content: Result Grid
94     where itemAmount>500);
95
96 -- Retrieve the productid,names of products whose quantity is 2
97 • select productid, name from products
98   where productid=( 
99       select productid from orderitem
100      where quantity=2);
101
102
103

```

productid	name
4	Headphones

Output:

Action	Time	Message	Duration / Fetch
102	21:54:07	selected p.* from products p where p.productid=(select o.productid from orderitem) LIMIT 0, 1000	Error Code: 1054. Unknown column 'o.productid' in field list
103	21:54:22	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	Error Code: 1242. Subquery returns more than 1 row
104	21:54:24	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	Error Code: 1242. Subquery returns more than 1 row
105	21:54:55	selected p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000	7 row(s) returned
106	21:57:19	select productid, name from products where productid=(select productid from orderitem where quantity=2)	0.000 sec / 0.000 sec

Delete order from orderitem whose productid is same as that in products table and having stock quantity is 5

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
106 -- DELETE
107 -- Delete order from orderitem whose productid is same as that in products table and
108 -- having stock quantity is 5
109
110 * delete from orderitem
111   where productid=(
112     select productid from products
113     where stockQuantity=5);
114 * select * from orderitem;
115
```

The results grid displays the following data:

orderItemID	productid	quantity	ItemAmount
1	2	1	1600
2	3	1	300
3	2	3	1800
4	4	1	600
5	6	1	50
6	1	1	800
7	2	2	1200
8	9	2	240
9	10	2	240

Output pane details:

- Action Output
- Time Action Message Duration / Fetch
- 104 21:54:24 select p.* from products p where p.productid=(select o.productid from orderitem o) LIMIT 0, 1000 Error Code: 1242: Subquery returns more than 1 row 0.000 sec
- 105 21:54:24 select p.* from products p where p.productid= ANY (select o.productid from orderitem o) LIMIT 0, 1000 7 rows(s) returned 0.000 sec / 0.000 sec
- 106 21:57:19 select p.productName from products p where productid=(select productid from orderitem where q.i=1) returned 0.000 sec / 0.000 sec
- 107 21:58:19 delete from orderitem where productid=(select productid from products where stockQuantity>5) 9 rows(s) affected 0.010 sec
- 108 21:58:34 select * from orderitem LIMIT 0, 1000 9 rows(s) returned 0.000 sec / 0.000 sec

Update product name whose product id is same as that in order item table having item amount is 1600.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and SQL Additions. The left sidebar displays the Navigator and Schemas (including career2, d, data, db, dd, demo, ecom, and others). The main area contains a query editor with the following code:

```
113 *      where stockQuantity=5);
114 *  select * from orderitem;
115 *
116 -- Update product name whose product id is same as that in order item table having item amount is 1600
117 * update products set name='Dell'
118 * where productids(
119 *     select productid from orderitem
120 *     where itemamount=1600);
121 *  select * from products;
122 *
```

The results grid shows the following data:

productID	name	Description	price	stockQuantity
1	Dell	High-performance laptop	\$800	10
2	Smartphone	Latest smartphone	\$600	15
3	Tablet	Portable tablet	\$300	20
4	Headphones	Noise-cancelling	\$150	30
5	TV	4K Smart TV	\$500	5
6	Coffee Maker	Automatic coffee maker	\$50	25
7	Refrigerator	Energy-efficient	\$700	10
8	Microwave Oven	Countertop microwave	\$80	15

The bottom pane shows the Action Output log with several entries, including:

- Time 21:57:19 Action select productid.name from products where productid= (select productid from orderitem where itemamount=1600)
Message 1 row(s) returned
- Time 21:58:19 delete from orderitem where productid= (select productid from products where stockQuantity=5)
Message 0 row(s) affected
- Time 21:58:34 select * from orderitem LIMIT 0, 1000
Message 9 row(s) returned
- Time 109 22:00:27 update products set name='Dell' where productid= (select productid from orderitem where itemamount=1600)
Message 0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
- Time 110 22:00:31 select * from products LIMIT 0, 1000
Message 10 row(s) returned

14) EXISTS operator

Retrieve Products which are placed as orders

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and a toolbar with various icons. The left sidebar displays the Schemas tree, which includes the `ecom` schema with tables like `customers`, `orderitems`, and `products`. The main area contains a SQL editor window with the following query:

```
72 * select * from orderitem;
73 * -- Retrieve Products which are placed as orders
74 * SELECT o.productID, Name
75 * FROM products p
76 * WHERE EXISTS (
77 *     SELECT 1
78 *     FROM OrderItem o
79 *     WHERE o.productID = p.productID
80 * )
81
```

Below the SQL editor is a Results Grid showing the following data:

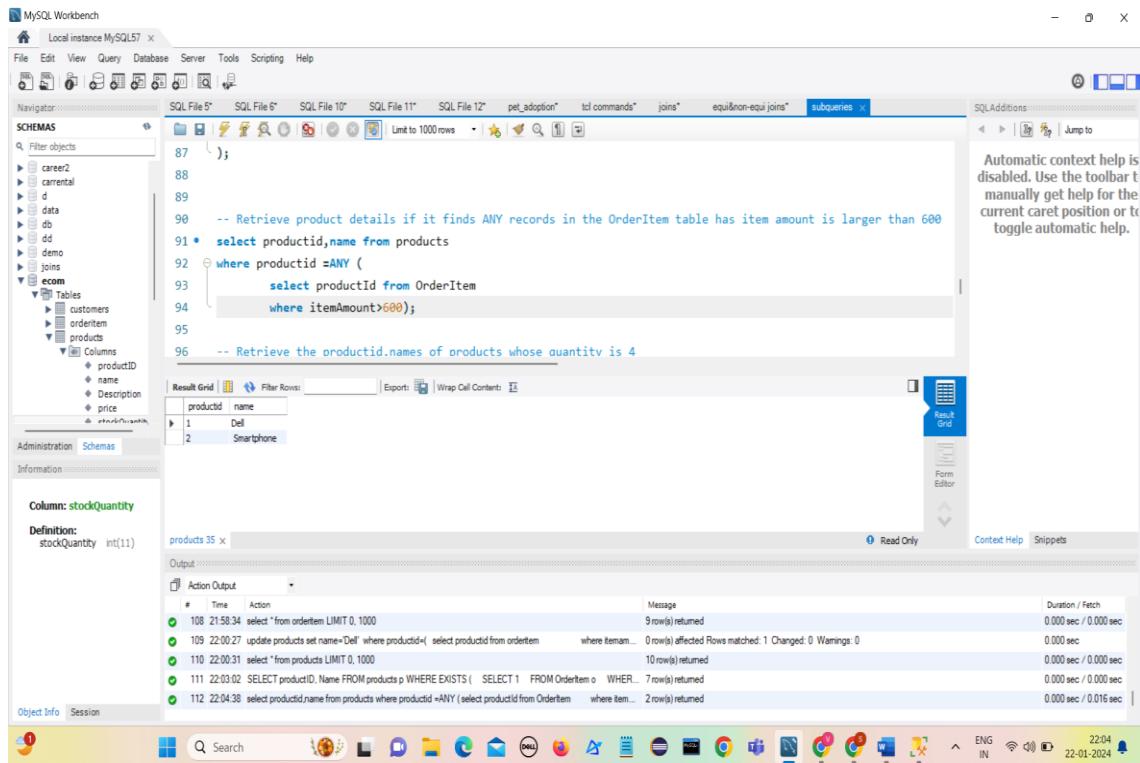
productID	Name
1	Smartphone
2	Tablet
3	Laptops
4	Coffee Maker
5	Blender
6	Household Cleaner
7	BBQ

The bottom pane shows the Output tab with the following log entries:

```
107 21:58:19 select * from orderitem; where productID=select productID from products
where stockQuantity>5) 0 rows affected
0 rows returned
108 21:58:34 select *from orderitem LIMIT 0, 1000
109 22:00:00 update product set name='Dell' where productID= select productID from orderitem
where itemname='Dell' 9 rows affected Rows matched: 1 Changed: 0 Warnings: 0
110 22:00:31 select *from products LIMIT 0, 1000
111 22:03:02 SELECT productID, Name FROM products p WHERE EXISTS ( SELECT 1 FROM OrderItem o WHERE
7 rows returned
```

15) ANY operator

Retrieve product details if it finds ANY records in the OrderItem table has item amount is larger than 600.



The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code in the editor is:

```
87    );
88
89
90 -- Retrieve product details if it finds ANY records in the OrderItem table has item amount is larger than 600
91 •  select productId,name from products
92   where productId =ANY (
93     select productId from OrderItem
94     where itemAmount>600);
95
96 -- Retrieve the productid.names of products whose quantity is 4
```

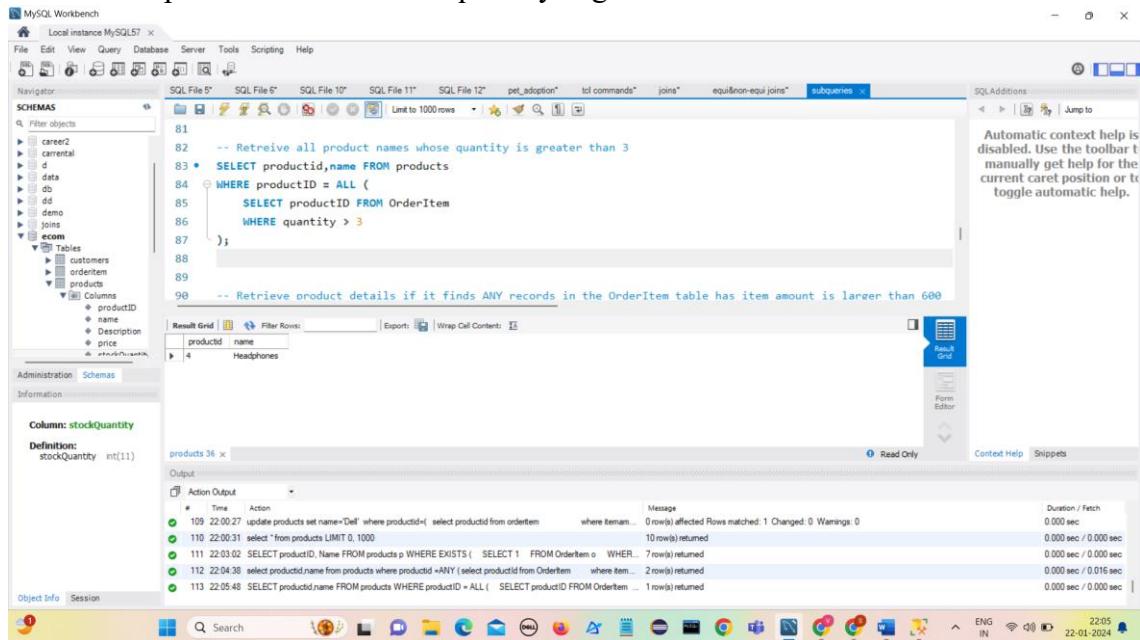
The results grid shows two rows of data:

productId	name
1	Dell
2	Smartphone

The status bar at the bottom right indicates the date and time as 22-01-2024 22:04.

16) ALL operator

Retrive all product names whose quantity is greater than 3



The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code in the editor is:

```
81
82 -- Retrive all product names whose quantity is greater than 3
83 •  SELECT productId,name FROM products
84   WHERE productId = ALL (
85     SELECT productId FROM OrderItem
86     WHERE quantity > 3
87   );
88
89
90 -- Retrive product details if it finds ANY records in the OrderItem table has item amount is larger than 600
```

The results grid shows one row of data:

productId	name
4	Headphones

The status bar at the bottom right indicates the date and time as 22-01-2024 22:05.

17) NESTED SUB QUERIES

A nested subquery is a subquery that is used inside another query. The inner query runs first, and the result of the inner query is used as the input for the outer query.

- 1) Retrieve customers who have placed orders for products with a price greater than the average price of all products

The screenshot shows the MySQL Workbench interface with a SQL editor window. The code is as follows:

```
128
129 •  SELECT * FROM customers
130 ◇ WHERE EXISTS (
131     SELECT 1 FROM OrderItem
132     WHERE productID IN (
133         SELECT productID FROM products
134         WHERE price > (
135             SELECT AVG(price) FROM products
136         )
137     )
138 );
139
140
```

The Result Grid shows the following customer data:

customerID	firstname	lastname	Email	address
1	John	Doe	john doe@example.com	123 Main St, City
2	Jane	Smith	jane smith@example.com	456 Elm St, Town
3	Robert	Johnson	robert j@example.com	789 Oak St, Village
4	Sarah	Brown	sarah b@example.com	101 Pine St, Suburb
5	David	Lee	daid l@example.com	234 Cedar St, District
6	Laura	Hall	laura@example.com	567 Birch St, County
7	Michael	Pearl	michael pearl@example.com	890 Maple St, State

The Output pane shows the execution log:

- 116 22:18:43 SELECT * FROM customers WHERE EXISTS (SELECT 1 FROM OrderItem WHERE productID IN (SELECT productID FROM products WHERE price > (SELECT AVG(price) FROM products))) returned 10 rows(s)
- 117 22:21:37 SELECT * FROM customers WHERE EXISTS (SELECT 1 FROM OrderItem WHERE productID IN (SELECT productID FROM products WHERE price > (SELECT AVG(price) FROM products))) returned 10 rows(s)

- 2) Names of products where the associated customer's first name starts with 'J'

The screenshot shows the MySQL Workbench interface with a SQL editor window. The code is as follows:

```
150 •  set sql_safe_updates=false
151 -- names of products where the associated customer's first name starts with 'J'.
152 ◇ SELECT name FROM products
153 ◇ WHERE productID IN (
154     SELECT productid FROM OrderItem
155     WHERE customerid IN (
156         SELECT customerID FROM customers
157         WHERE firstName LIKE 'J%'
158     )
159 );
160 •  select * from customers;
```

The Result Grid shows the following product data:

productID	name	Description	price	stockQuantity
5	4K Smart TV	900	5	10
6	Coffee Maker	Automatic coffee maker	50	25
7	Refrigerator	Energ-efficient	700	10
8	Microwave Oven	Countertop microwave	80	15
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10

The Output pane shows the execution log:

- 157 22:55:48 select * from products LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
- 158 22:56:05 select * from orderitem LIMIT 0, 1000 9 row(s) returned 0.000 sec / 0.000 sec
- 159 22:56:18 select * from products LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec

- 3) Retrieve the product names that have been ordered by customers with a specific address

```

MySQL Workbench
Local instance MySQL57 x
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* joins* equi&non-equi joins* subqueries* SQLAdditions
Schemas Filter objects
career2 currental d data dd demo joins ecom Tables customers customers products Columns productID name Description price
162 • select * from orderitem
163
164 -- Retrieve the product names that have been ordered by customers with a specific address
165 ② SELECT name FROM products
166 WHERE productID IN (
167   SELECT productID FROM OrderItem
168   WHERE customerID IN (
169     SELECT customerID FROM customers
170     WHERE address = '101 Pine St, Suburb'
171   )
172 )
Result Grid Filter Rows Export Wrap Cell Content
name
Tablet
products 69 x
Output
Action Output
# Time Action Message Duration / Fetch
162 22:59:49 select * from orderitem LIMIT 0, 1000 9 row(s) returned 0.000 sec / 0.000 sec
163 22:59:58 select * from products LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
170 23:01:08 SELECT name FROM products WHERE productID IN ( SELECT productID FROM OrderItem WHERE customerID = 1 ) 1 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

18) Correlated sub queries

SQL Correlated Subqueries are used to select data from a table referenced in the outer query.

The subquery is known as a correlated because the subquery is related to the outer query.

1) Write a correlated query to find the products with low stock.

```

MySQL Workbench
Local instance MySQL57 x
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* joins* equi&non-equi joins* subqueries & total aggregations* SQLAdditions
Schemas Filter objects
career2 currental d data dd demo joins ecom Tables customers customers products Columns productID name Description price
141
142
143 -- correlated queries
144 -- Write a correlated query to find the products with low stock
145 • select p.productID, p.name from products p
146 WHERE p.stockQuantity <
147   select avg(stockQuantity) from products
148 ;
149
Result Grid Filter Rows Export Wrap Cell Content
productID name
1 Dell
2 Smartphone
5 TV
7 Refrigerator
8 Microwave Oven
10 Vacuum Cleaner
products 106 x
Output
Action Output
# Time Action Message Duration / Fetch
219 09:37:06 select p.productID, p.name from products p where p.stockQuantity < ( select avg(stockQuantity) ) from products Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
220 09:37:23 SELECT productID, name FROM products p WHERE p.stockQuantity < ( SELECT MIN(stockQuantity) ) F... 0 row(s) returned 0.015 sec / 0.000 sec
221 09:37:33 SELECT productID, name FROM products p WHERE p.stockQuantity < ( SELECT avg(stockQuantity) ) F... 6 row(s) returned 0.000 sec / 0.000 sec
222 09:37:55 select productID, name from products p where p.stockQuantity < ( select avg(stockQuantity) ) from products Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
223 09:38:13 select productID, name from products p where p.stockQuantity < ( select avg(stockQuantity) ) from products 6 row(s) returned 0.000 sec / 0.000 sec
224 09:38:26 select p.productID, p.name from products p where p.stockQuantity < ( select avg(stockQuantity) ) from products 6 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

```

2) Write a correlated query to find products with price more than average.

```

MySQL Workbench - Local instance MySQL57

File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* joins* equi&non-equi joins* subqueries & total aggregations* SQL Additions
Filter objects
SCHEMAS
  career2
  currental
  d
  data
  db
  dd
  demo
  joins
  ecom
    Tables
      customers
      orderitem
        Columns
          orderItemID
          productID
          quantity
          itemAmount
          customerID
Result Grid Filter Rows Export Wrap Cell Content
149
150 -- Write a correlated query to find products with price more than average.
151 • select p.productid,p.name from products p
152   where p.price>
153     select avg(price) from products
154   );
155
156
157

Result Grid Filter Rows Export Wrap Cell Content
productID name
1 Dell
2 Smartphone
5 TV
7 Refrigerator

Output
Action Output
# Time Action Message Duration / Fetch
220 09:37:23 SELECT productID, name FROM products p WHERE p.stockQuantity < ( SELECT MIN(stockQuantity) FROM products ) 0 row(s) returned 0.015 sec / 0.000 sec
221 09:37:33 SELECT productID, name FROM products p WHERE p.stockQuantity < ( SELECT avg(stockQuantity) FROM products ) 6 row(s) returned 0.000 sec / 0.000 sec
222 09:37:55 select productID,name from products p where p.stockQuantity<( select avg(stockQuantity)from products ) Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
223 09:38:13 select productID,name from products p where p.stockQuantity<( select avg(stockQuantity)from products ) 6 row(s) returned 0.000 sec / 0.000 sec
224 09:38:26 select p.productID,p.name from products p where p.stockQuantity<( select avg(stockQuantity)from products ) LIMIT 0,... 4 row(s) returned 0.000 sec / 0.000 sec
225 09:41:46 select p.productID,p.name from products p where p.price<( select avg(price) from products ) LIMIT 0,... 4 row(s) returned 0.000 sec / 0.000 sec

Object Info Session

```

19) UNION, INTERSECT, EXCEPT

1) UNION

Retrieve a combined list of distinct products from the 'products' and 'OrderItem' tables

```

MySQL Workbench - Local instance MySQL57

File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands* joins* equi&non-equi joins* subqueries * SQL Additions
Filter objects
SCHEMAS
  career2
  currental
  d
  data
  db
  dd
  demo
  joins
  ecom
    Tables
      customers
      orderitem
      products
      orderitems
        Columns
          productID
          name
          Description
          price
Result Grid Filter Rows Export Wrap Cell Content
171
172   );
173
174 -- UNION
175 -- Retrieve a combined list of distinct products from the 'products' and 'OrderItem' tables
176 • SELECT productID, name FROM products
177 UNION
178 SELECT productID, '' FROM OrderItem;
179
180 -- INTERSECT & EXCEPT in SSMS
181

Result Grid Filter Rows Export Wrap Cell Content
productID name
1 Dell
2 Smartphone
3 Tablet
4 Headphones
5 TV
6 Coffee Maker
7 Refrigerator
8 Minnows Own

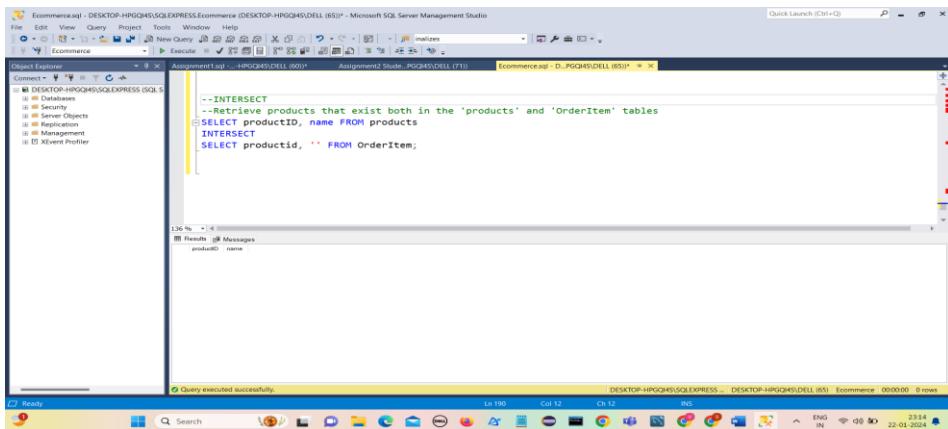
Output
Action Output
# Time Action Message Duration / Fetch
173 23:09:19 SELECT productID, name FROM products minus SELECT productID, '' FROM Orderitem Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
174 23:09:48 SELECT productID, name FROM products EXCEPT SELECT productID, '' FROM Orderitem Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se... 0.000 sec
175 23:13:16 SELECT productID, name FROM products UNION SELECT productID, '' FROM Orderitem 17 row(s) returned 0.000 sec / 0.000 sec

Object Info Session

```

2) INTERSECT

Retrieve products that exist both in the 'products' and 'OrderItem' tables



```
--INTERSECT
--Retrieve products that exist both in the 'products' and 'OrderItem' tables
SELECT productID, name FROM products
INTERSECT
SELECT productid, '' FROM OrderItem;
```

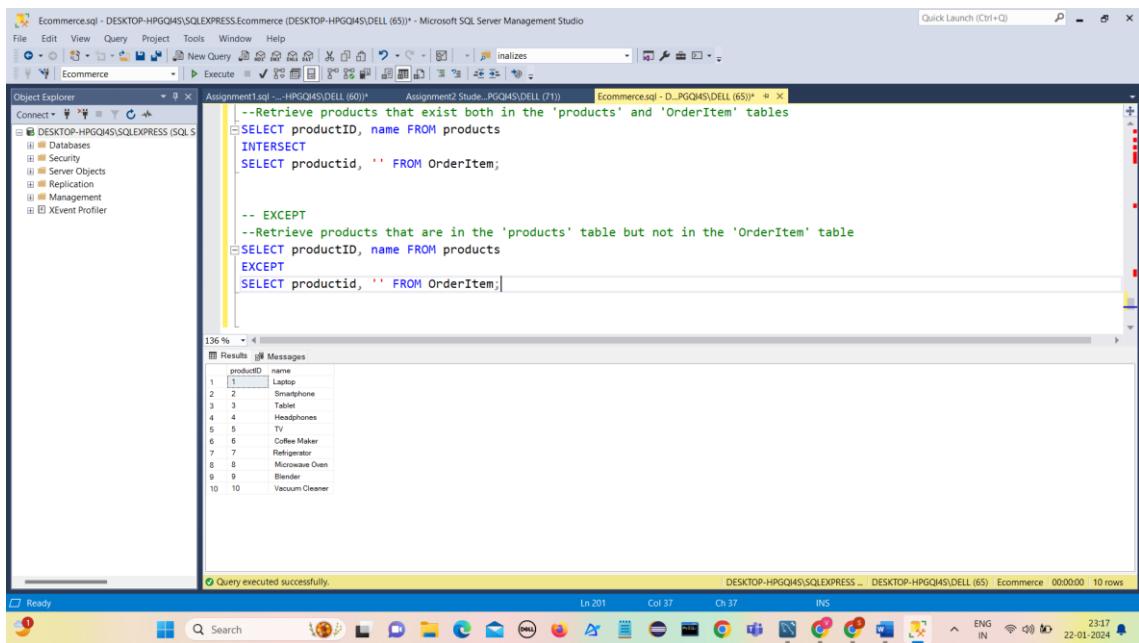
The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a connection to 'DESKTOP-HPGQ45\SQLEXPRESS'. In the center pane, a query window displays the above T-SQL code. Below it, the results pane shows a table with two columns: 'productID' and 'name'. The data returned is:

productID	name
1	Laptop
2	Smartphone
3	Tablet
4	Headphones
5	TV
6	Coffee Maker
7	Refrigerator
8	Microwave Oven
9	Blender
10	Vacuum Cleaner

At the bottom of the results pane, it says 'Query executed successfully.'

3) EXCEPT

Retrieve products that are in the 'products' table but not in the 'OrderItem' table



```
--INTERSECT
--Retrieve products that exist both in the 'products' and 'OrderItem' tables
SELECT productID, name FROM products
INTERSECT
SELECT productid, '' FROM OrderItem;

--EXCEPT
--Retrieve products that are in the 'products' table but not in the 'OrderItem' table
SELECT productID, name FROM products
EXCEPT
SELECT productid, '' FROM OrderItem;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a connection to 'DESKTOP-HPGQ45\SQLEXPRESS'. In the center pane, a query window displays the above T-SQL code. Below it, the results pane shows a table with two columns: 'productID' and 'name'. The data returned is:

productID	name
1	Laptop
2	Smartphone
3	Tablet
4	Headphones
5	TV
6	Coffee Maker
7	Refrigerator
8	Microwave Oven
9	Blender
10	Vacuum Cleaner

At the bottom of the results pane, it says 'Query executed successfully.'

4) MERGE

The **MERGE** statement in SQL is typically used for performing operations such as **INSERT**, **UPDATE**, or **DELETE** based on a condition.

Merge to update stock quantity in the products table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'Commerce' database is selected. In the center pane, a query window titled 'Assignment1.sql - not connected' contains the following SQL code:

```
MERGE INTO products AS target
USING (
    SELECT productID, SUM(quantity) AS totalQuantity
    FROM OrderItem
    GROUP BY productID
) AS source
ON target.productID = source.productID
WHEN MATCHED THEN
    UPDATE SET target.stockQuantity = target.stockQuantity - source.totalQuantity;

select * from products;
```

Below the code, the results of the query are displayed in a table:

productID	name	Description	price	stockQuantity
1	Laptop	High-performance laptop	800	7
2	Smartphone	Latest smartphone	600	10
3	Tablet	Portable tablet	300	19
4	Headphones	Noise-cancelling	150	26
5	TV	4K Smart TV	900	3
6	Coffee Maker	Automatic coffee-maker	50	24
7	Refrigerator	Energy-efficient	800	10
8	Microwave Oven	Countertop microwave	80	15
9	Blender	High-speed blender	70	17
10	Vacuum Cleaner	Bagless vacuum cleaner	120	8

At the bottom of the results pane, it says 'Query executed successfully.' The status bar at the bottom right shows 'DESKTOP-HPGQ4S\SOLEXPRESS ... DESKTOP-HPGQ4S\DELL (52) Ecommerce 00:00:00 10 rows'. The taskbar at the bottom of the screen shows various application icons.

Vuthur Sriganga
Hexaware Data Engineering Batch-1