BSCCS2003: Practice Questions with Solutions
Week 4

1. Consider the following two tables: [MCQ]

Table 1: Instructors

Instructor_ID	Instructor_Name
101	Amit Dubey
102	Sarthak Gaur
103	Neha Sharma
104	Sumit Kumar
105	Himanshi Mehra

Table 2: Teaches

Instructor_ID	Course_ID
102	CS-206
104	CS-208
103	CS-202
101	CS-206
105	CS-202

Which of the following queries will list the names of those instructors who teach a course having Course ID = “206”?

○ SELECT Instructor_Name FROM Instructors, Teaches
WHERE Course_ID = ‘CS-206’;

√ SELECT Instructor_Name FROM Instructors, Teaches
WHERE Instructors.Instructor_ID = Teaches.Instructor_ID
and Course_ID = ‘CS-206’;

○ SELECT Instructor_Name FROM Instructors
WHERE Instructors.Instructor_ID = Teaches.Instructor_ID
and Course_ID = ‘CS-206’;

○ All of the above

Solution:
Option 1: The first query will do a cartesian product and generate spurious information in result.
Option 2: The second query will result only in the names of those instructors who teach the course with Course_ID = ‘CS206’.
Option 3: The third query will generate an error as the table “Teaches” is not in FROM clause.

2. Consider the following table: [MCQ]

Table 1: Instructors

Instructor_ID	**Instructor_Name**
101	Amit Dubey
102	Sarthak Gaur
103	Neha Sharma
104	Sumit Kumar
105	Himanshi Mehra
106	Minakshi Mehta
107	Pushkar Kashyap
108	Siya Mishra

Which of the following queries will list the names of those instructors who have ‘i’ as the second alphabet and ‘r’ as the second last alphabet in their names?

√ SELECT Instructor_Name FROM Instructors
WHERE Instructor_Name LIKE ‘_i%r_’;

○ SELECT Instructor_Name FROM Instructors
WHERE Instructor_Name LIKE ‘%i%r%’;

○ SELECT Instructor_Name FROM Instructors
WHERE Instructor_Name LIKE ‘%i_r%’;

○ None of the above

Solution:
Option 1: The first query will result in the names of all those instructors who have ‘i’ as second alphabet and ‘r’ as second last alphabet in their names.
Option 2: The second query will result in the names of all those instructors whose name has alphabets ‘i’ and ‘r’.
Option 3: The third query will result in the names of all those instructors whose name has alphabets ‘i’ and ‘r’ separated by any character.

3. What will be the output of the following code snippet?

```
class Student:
    idnext=0
    def __init__(self, name):
        self.name=name
        self.idnext=Student.idnext
        Student.idnext=Student.idnext+1
s1=Student("Rohan")
s2=Student("Karthik")
s3=Student("Saurav")
print(Student.idnext)
```

[MCQ]

- ○ 0
- ○ 1
- √ 3
- ○ None of the above

Solution: The 'idnext' is a class variable. So it can be called directly from a class.

Whenever an object of the class Student is created, the value of the class variable idnext increases by 1.

4. Which of these programs is/are used to create spreadsheets? [MCQ]

- ○ Google Sheets
- ○ Microsoft Excel
- √ Both A and B
- ○ None of the above

Solution: These programs like Google sheets and Microsoft Excel are used to create spreadsheets. They are used by many businesses to manage, display, and manipulate data.

5. Which of the following statements is/are correct about CSV file in Python? [MCQ]

   √ A CSV file stores tabular data, in which each data field is generally separated by a delimiter (comma).

   ○ To save a CSV file, it must be saved with a **.csv** extension.

   ○ Both A and B

   ○ None of the above

Solution: CSV (Comma Separated Values) is a simple file format accustomed to store tabular data, like a spreadsheet or database.

6. Which of the following is/are valid join(s) in SQL? [MSQ]

   √ INNER JOIN

   √ FULL JOIN

   √ LEFT JOIN

   √ RIGHT JOIN

Solution: JOINS in SQL are commands which are used to combine rows from two or more tables, based on a common column between those tables.
There are mainly four types of joins: INNER JOIN, FULL JOIN, LEFT JOIN, RIGHT JOIN.

7. Which of the following is/are valid syntax(es) of the INNER JOIN? [MSQ]

√ Select table1.column1,
table2.column2
FROM table2
INNER JOIN table1
ON table1.common_field = table2.common_field;

○ SELECT table1.column2, table2.column2
From table1
Inner join table1
UPON table2.common_field = table2.common_field;

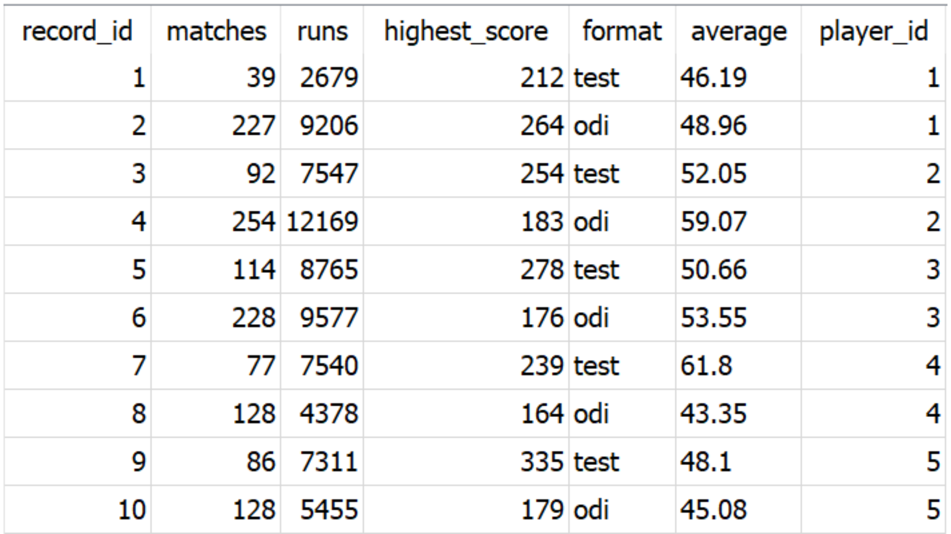
√ SELECT table1.column1,
table2.column2
FROM table1
INNER JOIN table2
ON table1.common_field = table2.common_field;

○ None of the above

Solution: Both option i and iii, are valid syntax of the INNER JOIN.
The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate.
When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

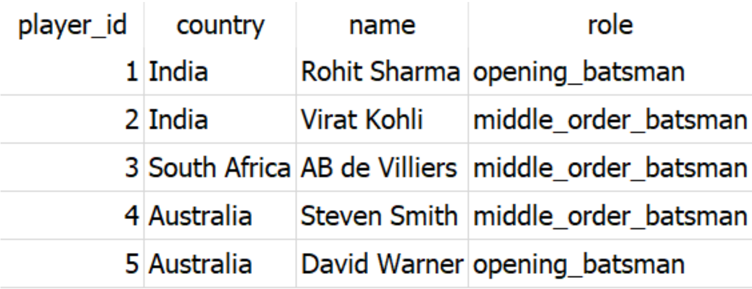
8. Consider the following tables. [MCQ]

1) Table 1: records



record_id	matches	runs	highest_score	format	average	player_id
1	39	2679	212	test	46.19	1
2	227	9206	264	odi	48.96	1
3	92	7547	254	test	52.05	2
4	254	12169	183	odi	59.07	2
5	114	8765	278	test	50.66	3
6	228	9577	176	odi	53.55	3
7	77	7540	239	test	61.8	4
8	128	4378	164	odi	43.35	4
9	86	7311	335	test	48.1	5
10	128	5455	179	odi	45.08	5

2) Table 2: players



player_id	country	name	role
1	India	Rohit Sharma	opening_batsman
2	India	Virat Kohli	middle_order_batsman
3	South Africa	AB de Villiers	middle_order_batsman
4	Australia	Steven Smith	middle_order_batsman
5	Australia	David Warner	opening_batsman

Which of the following queries will produce the name of the batsman with highest "highest_score" in odi?

√

```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
WHERE records.format = "odi"
ORDER BY records.highest_score DESC
LIMIT 1;
```

○

```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
WHERE records.format = "odi"
ORDER BY records.highest_score
LIMIT 1;
```

○

```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
ORDER BY records.highest_score
WHERE records.format = "odi"
LIMIT 1;
```

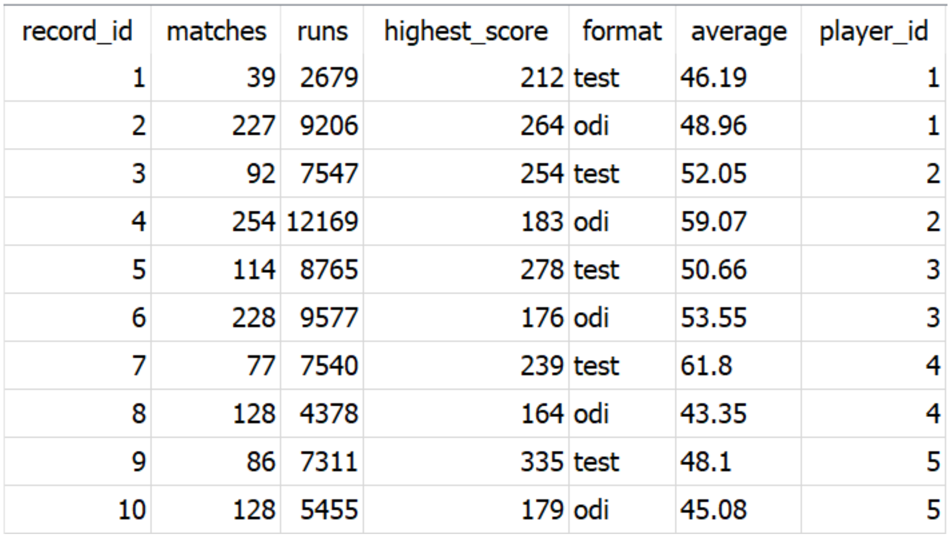
○

```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
ORDER BY records.highest_score DESC
WHERE records.format = "odi"
LIMIT 1;
```

Solution:
Option 1 will produce the name of the player with highest "highest_score" in odi.
Option 2 will produce the name of the player with lowest "highest_score" in odi.
Option 3 and 4 will produce an error because ORDER BY should come after the WHERE clause.

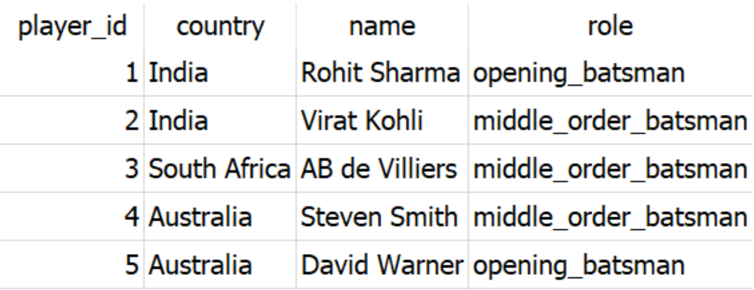
9. Consider the following tables. [MCQ]

1) Table 1: records



record_id	matches	runs	highest_score	format	average	player_id
1	39	2679	212	test	46.19	1
2	227	9206	264	odi	48.96	1
3	92	7547	254	test	52.05	2
4	254	12169	183	odi	59.07	2
5	114	8765	278	test	50.66	3
6	228	9577	176	odi	53.55	3
7	77	7540	239	test	61.8	4
8	128	4378	164	odi	43.35	4
9	86	7311	335	test	48.1	5
10	128	5455	179	odi	45.08	5

2) Table 2: players



player_id	country	name	role
1	India	Rohit Sharma	opening_batsman
2	India	Virat Kohli	middle_order_batsman
3	South Africa	AB de Villiers	middle_order_batsman
4	Australia	Steven Smith	middle_order_batsman
5	Australia	David Warner	opening_batsman

Which of the following queries will produce the name of the batsman with lowest "highest_score" in test?

○
```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
WHERE records.format = "test"
ORDER BY records.highest_score DESC
LIMIT 1;
```

√
```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
WHERE records.format = "test"
ORDER BY records.highest_score
LIMIT 1;
```

○
```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
ORDER BY records.highest_score
WHERE records.format = "test"
LIMIT 1;
```

○
```
SELECT players.name
FROM players
INNER JOIN records ON records.player_id = players.player_id
ORDER BY records.highest_score DESC
WHERE records.format = "test"
LIMIT 1;
```

Solution: Option 1 will produce the name of the player with highest "highest_score" in test matches.
Option 2 will produce the name of the player with lowest "highest_score" score in test matches.
Option 3 and 4 will produce an error because ORDER BY should come after the WHERE clause.

10. What will be the correct syntax to create a table called ‘Persons’ that contains five columns: PersonID, LastName, FirstName, Address, and City? [MCQ]

○
```
CREATE TABLE Persons (
    PersonID int
    LastName varchar(255)
    FirstName varchar(255)
    Address varchar(255)
    City varchar(255)
);
```

√
```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

○
```
CREATE TABLE (Persons) (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

○
```
CREATE TABLE AS Persons (PersonID, LastName, FirstName, Address, City);
```

Solution: The correct syntax to create a table in SQL is:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

In options I, III and IV, this syntax is violated.

11. The SQL syntax for BETWEEN operator is given below:

```
SELECT Prices
FROM Grocery
WHERE Prices BETWEEN 1000 AND 1500;
```

What will the code segment given above do? [MCQ]

○ It will select all the records from the 'Prices' column of table 'Grocery' between 1000 and 1500, both 1000 and 1500 exclusive.

✓ It will select all the records from the 'Prices' column of table 'Grocery' between 1000 and 1500, both 1000 and 1500 inclusive.

○ It will select all the records from the 'Prices' column of table 'Grocery' between 1000 and 1500 and is applicable only if values are numbers.

○ It will throw an error, as BETWEEN operator is applicable only if the values are dates.

Solution: The operator BETWEEN selects all the values from the given fields that is/are between the given range, with both the extreme values included. The values can be numbers, texts or dates.