

## **2013\_June**

### **A1.**

- a)- relation
- b)- tuple
- c)- column
- d)- field
- e)- database
- f)- select
- g)- from
- h)- column alias
- i)- arithmetic
- j)- single

### **A2.**

- a)- false
- b)- false
- c)- true
- d)- true
- e)- false
- f)- true
- g)- true
- h)- true

### **A3. Comparison operators**

**NOT**

**AND**

**OR**

**A4.**

```
SELECT GetDate() As Today  
FROM TEST;
```

**A5.**

```
Cstr()
```

**A6.**

```
SELECT item, grade*weight As price  
FROM stock;
```

**A7.**

- a)- \_
- b)- [charlist]
- c)- AND
- d)- OR
- e)- NOT
- f)- BETWEEN
- g)- IN
- h)- LIKE
- i)- IS NULL

**B1.**

a)

-

```
SELECT Student_Name, Student_Id
FROM STUDENT
WHERE Enrol_Date BETWEEN= '2008-01-01' AND '2008-12-31';
```

**OR**

```
SELECT Student_Name, Student_Id
FROM STUDENT
WHERE YEAR(Enrol_Date)= 2008;
```

b)

-

```
SELECT Student_Name, Subject_ID
FROM Student
WHERE Enrol_Date BETWEEN '2007-08-01' AND '2008-08-31';
```

c)

```
SELECT Student_Name
FROM Student
WHERE Mark > (SELECT Mark
              FROM Student
              WHERE Student_Name= 'Wendy');
```

d)

i.-

INSERT, UPDATE

ii.-

GRANT, REVOKE

e)

```
SELECT S.Sname, C.Module_Code
FROM STUDENT As S
INNER JOIN CLASSES As C ON S.Module_Code=C.Module_Code
WHERE C.Lecturer= 'Catherine';
```

## B2.

- a) 

```
SELECT Concat(Subject_Name, ' subject belongs to ',Department) As [Subject-Department]
FROM Subject;
```
- b) 

```
SELECT Scr.Student_Id, Sub.Subject_Name,Scr.Mark
FROM SUBJECT As Sub
INNER JOIN SCORE AS Scr ON Sub.Subject_Id=Scr.Subject_Id;
```
- c) - A Cartesian product is a join or combination of every row of one table to every row of another table. That is, if you have m rows in one table and n rows in the other, you get  $m \times n$  rows in the result.
- d)- 160 row.
- e) Page 11. | **Topic- 2.2**
- f)
- Objectives:
  - It is used to sort the data in ascending or descending order, based on one or more columns.
  - When TOP clause is specified, it serves to filter rows.

**B3.**

a)- `SELECT (5+219)*15;`

b)- All the names of the employee will be retrieved from the table **EMP** whose salary is between inclusive lowest and highest amount and doesn't contain null value.

c)- `SELECT COUNT(*)`  
`FROM EMPLOYEE`  
`GROUP BY DEPTNO;`

d)- `SELECT NAME, Format(PRICE, 'C') As [Price $]`  
`FROM BEVERAGE;`

e)-

- A primary key constraint enforces uniqueness of rows.
- It cannot contain *NULL* values.
- It must contain unique values.

f)- `SELECT *`  
`FROM CLIENT`  
`WHERE Cname LIKE 'Ke%';`