SOLUTIONS SQL WORKSHEET-6

1) A, C and D
2) A, C and D
3) C
4) C
5) D
6) B
7) A
8) C
9) D
10) A

11) **Denormalization** refers to the process of deliberately adding redundant data to a database to improve performance. This is in contrast to normalization, which involves dividing data into separate tables to eliminate redundancy and improve data consistency. Denormalization involves combining data from multiple tables into a single table, adding redundant data to improve query performance. This design trades off some of the benefits of normalization for improved performance, but also increases the risk of data anomalies and makes it more difficult to maintain data consistency. Denormalization is typically used in data warehousing, online transaction processing (OLTP) systems, and other database systems that require fast and efficient data retrieval, even if it comes at the cost of data redundancy and consistency.

12) A **cursor** is a control structure that enables traversal over the result set of a SELECT statement. In other words, it allows to traverse the rows of a result set one at a time, instead of retrieving all the rows at once. A cursor can be used to:
    - Retrieve a single row from the result set and process it
    - Modify the data of a single row and make changes to the underlying table
    - Fetch data in a specific order
    - Improve performance when dealing with large result sets

13) SQL (Structured Query Language) has several types of queries that can be used to perform various tasks in a database. Some of the most commonly used types of SQL queries include:
    - **SELECT** - retrieves data from one or more tables in the database.
    - **INSERT** - inserts new data into a table in the database.
    - **UPDATE** - updates existing data in a table in the database.
    - **DELETE** - deletes data from a table in the database.

- **ALTER** - alters the structure of a table in the database.
- **CREATE** - creates a new table, view, index, or other database object in the database.
- **DROP** - deletes a table, view, index, or other database object in the database.
- **INDEX** - creates an index on one or more columns in a table to improve query performance.
- **WHERE** - specifies conditions that limit the rows returned by a SELECT query.
- **GROUP BY** - groups rows in a SELECT query based on the values in one or more columns.
- **HAVING** - specifies conditions that filter the grouped rows in a SELECT query.
- **ORDER BY** - sorts the rows returned by a SELECT query in a specific order.

These are some of the most commonly used types of SQL queries, but there are many others as well, each with its own specific syntax and usage.

14) A constraint is a rule that restricts the values that can be stored in one or more columns of a table. Constraints are used to maintain the integrity of the data in the database. There are several types of constraints in SQL, including:
    - **NOT NULL constraint**: This constraint ensures that a column cannot store NULL values.
    - **UNIQUE constraint**: This constraint ensures that the values in a column are unique, i.e., no two rows can have the same value in a column with a UNIQUE constraint.
    - **PRIMARY KEY constraint**: This constraint is used to uniquely identify each row in a table. A table can have only one primary key, and it must consist of one or more columns.
    - **FOREIGN KEY constraint**: This constraint is used to establish a relationship between two tables. It ensures that the values in a column of one table match the values in a column of another table.
    - **CHECK constraint**: This constraint restricts the values that can be stored in a column by evaluating a Boolean expression. If the expression evaluates to false, the constraint is violated and the insert or update operation is not allowed.
    - **DEFAULT constraint**: This constraint assigns a default value to a column when no value is specified during an insert operation.

15) **"Auto increment"** is a feature that automatically generates unique integer values for a column in a table. It is commonly used as a primary key for a table, as primary keys must

be unique and not null. The auto-increment feature allows the database to automatically assign a unique value to the column for each new record inserted into the table, without the need for the user to manually specify the value.

For example, in a table named "users" with an auto-increment primary key named "id", the following SQL statement can be used to create the table:

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR (255),
  email VARCHAR (255)
);
```

Then, when a new user is inserted into the table, the "id" column will be automatically assigned a unique value:

```
INSERT INTO users (name, email) VALUES ('John Doe', 'john.doe@example.com');
```

In this example, the "id" column will be automatically set to 1 for the first record, 2 for the second, and so on.