Advanced Queries and Joins

Sr. No.	Assignment Question

1. **Cosmos Electronics Ltd.** employs more than 1,000 workers in its units. Some of these are at junior level while some are at senior level depending upon their expertise and years of experience. Each employee is given annual leave based on the designation. The management at **Cosmos Electronics Ltd.** is planning to computerize their human resources department and all the data pertaining to employees will now be stored in SQL Server 2012 databases. The structure for **EmpDetails** and **LeaveDetails** are shown in tables 9.1 and 9.2 respectively.

   - **EmpDetails Table**

Field Name	Data Type	Key Field	Description
Emp_Id	varchar(5)	Primary Key	Stores employee identification number
FirstName	varchar(30)		Stores first name of the employee
LastName	varchar(30)		Stores last name of the employee
Address	varchar(60)		Stores address of the employee
PhoneNumber	varchar(20)		Stores phone number of the employee, it could be landline or mobile
Dept_Id	varchar(4)		Stores department id of the department to which the employee belongs
Designation	varchar(30)		Stores designation or job role of the employee
Salary	money		Stores salary of the employee
Join_date	datetime		Stores date of joining for the employee
Performance_Rating	int		Stores Rating of the employee

Table 9.1: EmpDetails Table

- **LeaveDetails Table**

Field Name	Data Type	Key Field	Description
Emp_Id	varchar(5)	Primary Key	Stores employee identification number
LeaveTaken	int		Stores the number of leaves taken by the employee
FromDate	datetime		Date from when leave was availed
ToDate	datetime		Date upto which leave was taken
Reason	xml		Reason for the leave

Table 9.2: LeaveDetails Table

1. Using SQL Server 2012 and Transact SQL statements create the tables in a database named **Cosmos**. Add at least five records to each table.

2. Create a query to retrieve the number of leaves taken by employees having designation as Manager.

3. Create a query to retrieve details of all employees who have taken leave for more than five days.

4. Compare the records of employees with each other to see who have taken more number of leaves (**Hint**: use a self-join on the **LeaveDetails** table).

Using Views, Stored Procedures, and Querying Metadata

Sr. No.	Assignment Question

1. **BookParadise** is an online library management system used by a library in **Seattle**. The software makes use of SQL Server 2012 databases. Information about thousands of books are maintained and updated regularly. In the recent years, **BookParadise** has grown in size after receiving international funding and the number of books has increased tremendously. This increase in the number of books has made searching for books very difficult. Also, **BookParadise** needs to now allow searches to be made by users displaying only selective information to them. The entire content of the tables are not to be displayed to the users. Towards this end, views will be created to enable the readers to display information about books, in a fast and efficient manner.

   The **Books** table in the **BookParadise** database has the structure shown in table 10.1.

   - **Books**:

Field Name	Data Type	Key Field	Description
BookCode	varchar(5)	Primary Key	Book Identification Code
Title	varchar(30)		Title of the book
Author	varchar(30)		Author of the book
Edition	int		Edition number
RatePurchased	money		Cost price of the book
PurchaseDate	datetime		Date when book was purchased
VendorName	varchar(30)		Vendor who sold the book
BookStatus	varchar(15)		Indicates whether book is available or not

Table 10.1: Books Table

All the above fields, except the primary key, may accept null values.

1. Using SQL Server 2012 and Transact SQL statements, create the above table in a database named **BookParadise**. Add at least seven records to the table.

2. Next, create a view named **BookInfo** on the table, which will contain columns BookCode, Title, Author, Edition, and BookStatus.

3. Test the view by displaying information from it. Display all the records in the view. Also, display the top 3 records in the view alphabetically sorted by the column Author.

4. Add three more records to the view.
5. Update all author names to replace space with '-'

2. **ToyzUnlimited** is a trendy toy store based in California. It buys toys from manufacturers, stocks them in its store, and sells them for profits. **ToyzUnlimited** maintains the details of all branded toy products in a SQL Server 2012 database. To speed up the day-to-day tasks and operations related to the database, it has been decided to use SQL Server 2012 stored procedures for commonly performed tasks. The structure for **Toys** table is shown in table 10.2.

   - **Toys**:

Field Name	Data Type	Key Field	Description
ProductCode	varchar(5)	Primary Key	Product Code that uniquely identifies each toy
Name	varchar(30)		Name of the toy
Category	varchar(30)		Category of the toy. Example: Block building, Board games, Puzzles, and so on.
Manufacturer	varchar(40)		Manufacturer name
AgeRange	varchar(15)		Age range for the kids to use the toy. Example: 3-5 years
UnitPrice	money		Price of the toy in dollars
Netweight	int		Weight of the toy in grams
QtyOnHand	int		Quantity available

Table 10.2: Toys Table

1. Start with creating the table **Toys**. The structure of the table is described above. Add at least five records to the table. Ensure that the value of the column QtyOnHand is more than 20 for each of the toys.

2. Write statements to create a stored procedure named **HeavyToys** that will list names of all the toys that are above 500 grams.

3. Write statements to create a stored procedure named **PriceIncrease** that will increment the unitprice of all toys by 10 dollars.

4. Write statements to create a stored procedure **QtyOnHand** that will decrease the quantity on hand of all toys by five.

5. Execute the stored procedures **HeavyToys**, **PriceIncrease**, and **QtyOnHand**.

6. Change the procedures **PriceIncrease** and **QtyOnHand** such that they will also display the newly updated values of price and quantity after the updation of these columns have taken place through the stored procedures mentioned earlier.

7. Remove all the stored procedures that were created so far.