FIT1013 – Digital Futures: IT for Business
Tutorial 11 – Querying a Database and Creating Advanced Queries

Objectives
- Create a query based on multiple tables
- Use a comparison operator in a query to match a range of values
- Use the And and Or logical operators in queries
- Create and format a calculated field in a query
- Perform calculations in a query using aggregate functions and record group calculations
- Use the Like, In, Not, and & operators in queries
- Create a parameter query
- Use query wizards to create a crosstab query, a find duplicates query, and a find unmatched query
- Create a top values query

Exercise 1
Download the Access file Task.accdb

GoGopher! Amol Mehta, a recent college graduate living in Boulder, Colorado, spent months earning money by running errands and completing basic chores for family members and acquaintances, while looking for full-time employment. As his list of customers needing such services continued to grow, Amol decided to start his own business called GoGopher! The business, which Amol operates completely online from his home, offers customers a variety of services-from grocery shopping and household chores to yard work and pet care-on a subscription basis. Clients become members of GoGopher! by choosing the plan that best suits their needs. Each plan provides a certain number of tasks per month to members for a specified time period of time. Amol created an Access database named Task to store data about members, plans, and contracts. He wants to create several new queries and make design changes to the tables. Complete the following steps:

1. Open the Task database.

2. Modify the first record in the tblMember table datasheet by changing the First Name and LastName column values to your first and last names. Close the table.

3. Create a query to find all records in the tblPlan table in which the PlanCost field is 600, 900, or 1500. Use a list-of-values match for the selection criterion, and include all fields from the table in the query recordset. Sort the query in descending order by the PlanID field. Save the query as **qryLowVolumePlans**, run the query, and then close it.

4. Make a copy of the qrylowVolumePians query using the new name **qryHighVolumePlans**. Modify the new query to find all records in the tblPlan table in which the PlanCost field is not 600, 900, or 1500. Save and run the query, and then close it.

5. Create a query to display all records from the tblMember Member table, selecting the LastName, FirstName, Street, and Phone fields, and sorting in ascending order by LastName and then in ascending order by FirstName. Add a calculated field named MemberName as the

first column that concatenates FirstName, a space, and LastName. Set the Caption property for the MemberName field to Member Name. Do not display the LastName and FirstName fields in the query recordset. Create a second calculated field named Cityline, inserting it between the Street and Phone fields. The City line field concatenates City, a space, State, two spaces, and Zip. Set the Caption property for the Cityline field to City Line. Save the query as qryMemberNames, run the query, resize all columns to their best fit, and then save and close the query.

6. Create a query to display all matching records from the tblPlan and tblMember tables, selecting the LastName and FirstName fields from the tblMember table and the Plan Description and PlanCost fields from the tblPlan table. Add a calculated field named FeeStatus as the last column that equals *Fee Waived* if the FeeWaived field is equal *to yes,* and that equals *Fee Not Waived* otherwise. Set the Caption property for the calculated field to Fee Status. Sort the list in ascending order on the LastName field. Save the query *as* qryFeeStatus, run the query, resize all columns to their best fit, and then save and close the query.

7. Create a query based on the tblPlan and tblMember tables, selecting the LastName, FirstName, and City fields from the tblMember table and the FeeWaived, Plan Description, and PlanCost fields from the tblPlan table. The query should find the records in which the City field value is Boulder or Erie and the FeeWaived field value is *Yes.* Save the query as **qryBoulderAndErieFeeWaived**. Save and run the query, and then close the query.

8. Create a parameter query to select the tblMember table records for a City field value that the user specifies. If the user doesn't enter a City field value, select all records from the table. Display all fields from the tblMember table in the query recordset. Save the query as **qryMemberCityParameter**. Run the query and enter no value as the City field value, and then run the query again and enter **Boulder** as the City field value. Close the query.

9. Create a find duplicates query based on the tblMember table. Select Expiration as the field that might contain duplicates, and select all other fields in the table as additional fields in the query recordset. Save the query as **qryDuplicateMemberExpirationDates**, run the query, and then close it.

10. Create a find unmatched query that finds all records in the tblMember table for which there is no matching record in the tblPlan table. Select FirstName, LastName, and Phone fields from the tblMembers table. Save the query as **qryMembersWithoutPlans**, run the query, and then close it.

11. Create a new query based on the tblMember table. Display the FirstName, LastName, Phone, Expiration, and PlanID fields, in this order, in the query recordset. Sort in ascending order by the Expiration field, and then use the Top Values property to select the top 25 percent of records. Save the query as **qryUpcomingExpirations**, run the query, and then close it.

12. Use the Input Mask Wizard to add an input mask to the Phone field in the tblMember table. Create the input mask such that the phone number is displayed with a dot separating each part of the phone number. For instance, if the phone number is (303) 123-4567 it should be displayed as 303.123.4567 for new entries. Test the input mask by typing over an existing

Phone column value, being certain not to change the value by pressing the Esc key after you type the last digit in the Phone column, and then save and close the table.

13. Define a field validation rule for the PlanCost field in the tblPlan table. Acceptable field values for the PlanCost field are values greater than 500. Enter the message Value must be greater than 500 so it appears if a user enters an invalid PlanCost field value. Save your table changes, and then test the field validation rule for the PlanCost field; be certain the field values are the same as they were before your testing, and then close the table.

14. Define a table validation rule for the tblMember table to verify that Datejoined field values precede Expiration field values in time. Use an appropriate validation message. Save your table changes, and then test the table validation rule, making sure any tested field values are the same as they were before your testing.

15. Add a Long Text field named **MemberComments** as the last field in the tblMember table. Set the Caption property to **Member Comments** and the Text Format property to Rich Text. In the table datasheet, resize the new column to its best fit, and then add a comment in the Member Comments column in the first record about a job you would do for someone else, formatting the text with blue, italic font. Save your table changes, and then close the table.