UNIVERSITY of York
Information Services

Essential Access

Exercises

IT

www.york.ac.uk/it-services/training

Essential Access Tasks

Exercises to accompany Essential Access Course

Books 1 & 2

Sample files for use with these exercises can be found on the shared Teaching (T:) drive, accessible from any classroom PC and other centrally managed computers. The folder needed is:

Teaching (T:)\IT Training\Essential Access

Users cannot save to this drive, so you will need to copy the entire **Essential Access** folder to your own documents location.

The sample file to be used is indicated at the start of each set of exercises.

Version: October 2018

1 ~ Understanding Databases

Open the Access database **Student Records 1.accdb** for these exercises

1 Open the **Students** table in datasheet view and enter the following new student either by navigating to the empty field or using the 'New record' button. Note the pencil symbol when editing:

studentID	forename	surname	email
10510	Paul	Hitch	p.hitch@gmail.co.uk

2 Enter yourself as a new student. Save by moving to a different record or using the Save button.

If you left the **studentID** field empty you will receive a warning message. This is because **studentID** is the primary key and must be included in any record.
Either enter a made-up **studentID** (10505) or delete the record.

3 Find student number **10330**, Tara Maples. Correct her email to **yahoo.com**.

4 Close **Students** and open the **Modules** table. Find the Plasma Physics module (**moduleID** PHY201) and set the **roomID** to G/A/114.

5 Correct the spelling of POL103 to 'War and Peace'.

Extension

6 Prof Leo Richards is the new tutor for the module CSC203, AJAX Development. Locate his TutorID (**Tutors** table) and update the tutorID for this module in the **Modules** table.

7 Joseph Smith has received a mark of 79 in Nuclear Chemistry. Enter this into the **ModuleChoices** table.

2 ~ Query Essentials

Continue using the Access database **Student Records 1.accdb** for these exercises

1 Create a new query in Design view. Add the **Students** table. You may want to resize the table so you can see all the fields listed.

Add **studentID**, **forename** and **surname** to the grid.
(Try using both the double-click and drag-and-drop methods and see which you prefer)

2 Use the View button to switch to Datasheet view. Find student 10091 and correct her name to Suzy Watson.

Note: Even though queries do not store data, the results of a query are from the underlying table, and a change made here is changing the data stored within the table.)

3 Switch back to Design view. Apply a Sort to order surnames alphabetically and check the results in the datasheet view.

4 Amend your query to sort first by surname, then by forename.

(Hint: Access applies sort orders from left to right. You will need the surname field to appear first in the QBE grid)

5 The field **currentYear** indicates the year of study of students. Add **currentYear** to your query and apply a filter to only show only 1st years.(170)

6 Modify the query to only show 1st years with a surname “Jones” (4)

7 Save the query as **qryFirstYears**

8 Create a new query based on the **Students** table that will show the names and ID of all female 2nd year students. (93)

9 Edit the query design to show female 2nd year students with the surname “Chapman”. (2)

10 Save the query as **qrySecondYears**

Extension

11 Create a new query that will display the studentID, name and date of birth for all students born on 26/09/1995. (1)

12 Modify the query to find students born between 01/01/1995 and 31/12/1995. (57)

13 Modify the query again to only show 1st years born in 1995. Also include their email address and sort by surname. (24)

14 Save the query as **qryDob**

15 Create a query to show all 2nd and 3rd year students. (330)

2.1 - Multi-table queries

Continue using the Access database **Student Records 1.accdb** for these exercises

1 We want to view information about the modules the students are taking. This will require data from two related tables, **Students** and **ModuleChoices.** Both tables contain **studentID** fields which are linked with a 1-to-many relationship.

Create a new query and add *only* the **Students** table. Add **studentID** and both name fields; view the datasheet. (approximately 500 records)

2 Add the table **ModuleChoices** and view the datasheet again.

There are about 3 times as many results as before. This is because each student studies several modules and is listed once for each module they take.
Add the **moduleID** field and view the datasheet again.

3 Return to the design and filter the results to show details for **studentID** 10175. (David McDonald)

4 It would be helpful to see the module titles, and you may also need to view other module information. To do this we need to add a third table to our query.

Add the **Modules** table to the design and include the **moduleName** and **credits** fields.
View the datasheet again.

5 Clear the studentID from the criteria and save the query as **qryModules**.

Extension

6 Create a new query that will list each module and the tutor who teaches it. Show the module ID, module name and term along with the Tutor's ID, full name and email. (44)

7 Modify the design to display only History modules.

(Hint: Use wildcards to find only the modules which start with HIS. (6)

8 Modify the query to show all modules again. Add the room ID and capacity. You will need to add a 3rd table to do this. (40)

9 Why do you think adding the rooms table to the query has reduced the number of records returned? Looking carefully at the Modules table may help.<=

10 Modify your query to show the college that the rooms are in.

11 Save the query with a suitable name.

3 ~ Editing Table Design

Open the Access database **Student Records 2.accdb** for these exercises.

1 Open the **Modules** table in design view. Examine the fields in this table, paying attention to the data type of each, and the different field properties available.

2 Set the field size of **moduleID** to prevent codes longer than 6 characters from being entered.

3 The **departmentID** is always a 3 character code. Change the field size, and format it to always display in uppercase.

4 Some modules are compulsory for all students. Add a new field called **compulsory** with **Yes/No** data type to record this. Set the default value to **Yes**.

5 There are never more than 100 credits awarded for a module. Adjust the number field size to the most appropriate type. Set a validation rule to prevent entries higher than 100 and enter suitable validation text.

(Hint: See Number field properties on p 17 and Other useful properties on p 18)

6 Add a new field **modCreated** to record the date a new module is entered into the database. Format as Short Date and set the default value to be the current date.

7 Add a new field **tutorID** to store which tutor takes this module. Set as **Long Integer**.

8 Save your changes to the table design and switch to datasheet view. Pay attention to the warning messages. Enter an imaginary new module. Try entering:

- a **departmentID** longer than 3 characters
- **credits** greater than 100
- The changes you made should prevent this. You should see your default values in the **modCreated** and **compulsory** fields.

A new table

9 Create a new table in design view called **Tutors**. Add the following fields:

Field Name	Data Type	Field properties
tutorID	AutoNumber	Primary Key
title	Short Text	field size 10
firstname	Short Text	field size 50
lastname	Short Text	field size 50
dateOfBirth	Date/Time	format as short date

10 Switch to datasheet view and enter a new tutor. Note **tutorID** is automatically completed when you start entering data in any other fields.

3.1 - Relationships

Use the Access database **Student Records 3.accdb** for these exercises.

1 Open the **Relationships** window from the Database Tools tab. This isn't currently showing all tables in the database. Add the **Departments** and **Tutors** tables.

2 Create a relationship between **tutorID** (Tutors table) and **tutorID** (Modules table) and enforce referential integrity.

3 Create a relationship between **deptID** (Departments table) and **departmentID** (Modules table) and enforce referential integrity. Close the Relationship window, saving the changes.

4 The Politics department have changed and are now the department of Politics, Economics and Philosophy.

Open the Departments table in Datasheet view and attempt to change deptID from POL to PEP. If you set up the relationship correctly you will receive an error.
(Note: This is because there are related records in the Modules table that have the department code POL. These records would become 'orphaned' if the corresponding POL name was changed in the Departments table.)
Use escape to undo the update.

5 Attempt to delete the Politics department. This will also fail for similar reasons. Close the table.

6 Open the **Relationships** window. Modify the relationship between Modules and Departments to **Cascade Update** and **Cascade Delete**. Close the window.

7 Open the **Departments** table and attempt to change POL to PEP again. You should receive no errors. Open the Modules table. Politics modules will now show this new PEP department code.

(Note: This is due to the 'Cascade updates' option)

8 Return to the **Departments** table and delete the Politics department. You will receive a warning that because of the cascade delete this will not only delete Politics from the departments table but also delete all related records in other tables. Choose **Yes** and return to the modules table to confirm that all Politics modules have now gone.

(Note: cascade delete should be used very carefully)

3.2 - Importing data

Use the Access database **Student Records 2.accdb** for these exercises.

1 The **Departments.xlsx** Excel file in the Essential Access folder contains department information. Import this file into a new table. A unique 3 character department code is included in this spreadsheet. Select this as the primary key rather than letting Access add its own. Name this new table **Departments**.

2 Open the newly imported **Departments** table in design view. The field names all contain spaces which are normally best avoided. Rename the fields:

   a) Department Code -> deptID

   b) Department Name -> deptName

   c) Head of Department -> headOfDept

   d) Set the field size of deptID to prevent new entries being longer than 3 characters.

3 Close and save the changes to **Departments**.

Extension

4 The **RoomsAndColleges**.**accdb** Access file in the Essential Access folder contains a table of room information and a table of college information. Import both these tables.

5 It is possible to import other Access database objects, such as Forms, Reports and Queries. Import **qryLargeRooms** from **RoomsAndColleges.accdb**

6 Modify **qryLargeRooms** to show only the large rooms on the East campus. (10)

7 **NewStudents.txt** contains a list of students. Append these to the **Students** table. Check the Students table to see how the data looks. Open the text file to see why the data has not imported correctly.

4 ~ Creative Queries

4.1 - Calculated fields in Queries

Open the Access database **Student Records 4.accdb.**

Note: You will need to enable active content for some of the exercises.

We want a list of some students showing their module result and their name displayed sensibly.

1 Create a new query using **Students**, **ModuleChoices** and **Modules**. Include the fields: **studentID**, **forename, surname, modName**, **mark** (1502)

2 Add the necessary extra fields and criteria to show only modules from the History department (departmentID: HIS) running in the first term (56).

3 Save the query as **qryResultsList**.

4 The mark is out of 120. Add a calculated field called **adjustedMark** to show the mark as a percentage. Set this to display 1 decimal place.

(Tip: divide the mark by 120 and then use the field properties to set the format of the Adjusted Mark field to be Percent.)

5 Modify the query design so the records are listed in descending order by adjusted mark (range is 80.8% to 23.3%).

6 Add a new field, **fullName** and use concatenation to join the students' forename and surname together in this field, ensuring a space is included. Position the field to display as the first column on the left.

Extension

7 Add sorting to the query design to display the results ordered first by surname, then by forename and hide the forename and surname fields from the result.

8 Modify the **fullName** calculated field so names are displayed in the format *Surname, Initial* (eg Smith, J)

4.2 - Grouping and Totals

Continue using the Access database **Student Records 4.accdb.**

You have been asked to provide data on the average marks for modules.

1 Create a new query to show the module name (**Modules**) and mark (**ModuleChoices**). Enable the **Totals** row, and configure to calculate the average of the marks (to 2 decimal places). Save query as **qryAverageMarks**.

2 Modify the query to show average marks by department rather than module. You will need to add another table in order to include the department name.

3 In place of averaged mark field, configure a calculated field that averages the adjusted mark as a percentage (marks are out of 120), shown to 1 decimal place.

4.3 - Parameter queries

Continue using the Access database **Student Records 4.accdb.**

We want to be able to view data about a particular student.

1 Create a new query from the **Students** table including **studentID**, **forename**, **surname**, **currentYear** and **email**.

2 Change this into a parameter query that requests a student surname when run. Save the query as **qryChooseStudent** and close. Run the query and choose to show students with the surname Smith. (3)

3 Modify the query to request also the current year of the students. Run the query and find all the **Wilson**s in year 2. (2)

Can you also find **Wilson**s in all years using this parameter query?

4.4 - Outer Joins

Continue using the Access database **Student Records 4.accdb.**

A list of the capacity of rooms used for teaching modules is required.

1 First create a new query to list **moduleID**, **modName** from the **Modules** table. View results and note the number of records (44). Save the query as **qryModules.**

2 We now want to see the room capacities for all these modules. Add the **Rooms** table to the query and include the **roomID** and **capacity** field. Run the query and note the number of records (39).

3 The discrepancy arises because modules that have no room allocated are not included. Modify the join between **Modules** and **Rooms** to be an outer join showing all records from Modules. Run the query again and note that all modules (44) are now included, with gaps where no room has been allocated.

4 Modify the outer join to show all records from Rooms. Note the number of records (116). Open the Rooms table and note the total number of rooms (100).

Why are these numbers different? (sorting the query by **roomID** may help.)

4.5 - Action queries

Continue using the Access database **Student Records 4.accdb.**

At the end of each year there are two important data changes to make:

- Yr 3 students need transferring to a separate **Alumni** table
- Yr 1 and 2 students need promoting a year

These can be carried out using action queries.

1 Create a new Select query showing **studentID**, **forename**, **surname**, **email** and **currentYear** from the Students table. Configure a filter to display only 3rd years.

2 Modify this to a **Make Table** query. Set the table name to **Alumni**.

3 We don't need **studentYear** in our alumni table; modify the query so this field is not included in the display, but is still used in the filter.

View the datasheet and check the results look OK. (151)

4 Save the query as **qryMakeAlumniTable** and close it.

5 Run this **Make Table** query. Note the warning message. (If you cannot run the query, you may not have enabled active content when you opened the database)

This has made a copy of all 3rd years into a new **Alumni** table, but it has not removed them from the **Students** table.

6 Make a new **Delete** query to remove all 151 3rd years from **Students**. Check your query is selecting the correct students.

It won't work, but run the query to see the error message. You are unable to delete these students because there are still entries for them in **ModuleChoices**. Removing the students would break **referential integrity**.

7 Save the query as **qryDelete3rdYears** and close it.

8 Select **Database Tools > Relationships** and modify the relationship between **Students** and **ModuleChoices** to **cascade delete**. Close the relationships window, saving changes.

9 Run **qryDelete3rdYears** again and then view the **Students** table to confirm all 3rd Years are deleted. (349)

Promotion

10 Create a new query that selects all 2nd years (179). Change this to an **Update** query to change all 2nd years to 3rd years. Run the query and look in **Students** to confirm it worked. Why must you promote 2nd years before 1st years?

11 Modify the query so it changes all 1st years to 2nd years and run it again.

For the future…

12 Create an **Append** query to add all current 3rd years to the **Alumni** table. You can only include fields that exist in the Alumni table. (Tip: save a copy of **qryMakeAlumniTable** and change this to an **Append** query).

5 ~ Data Design: Rent-A-Toy

Rent-A-Toy is a service run by a local community centre. They own a collection of toys which local residents may borrow. New toys are purchased regularly and old ones reaching end of life are decommissioned. The loans are to be managed using an Access database system.

- Parents must register, providing contact details including address, phone and email (if they have one) and must be aged over 18
- Each toy has a suggested age range to help parents choose
- Toys are borrowed initially for 2 weeks, which can be extended by a further 2 weeks only once
- Toys are purchased with council funds, so statistics on the service need to be generated once a month, including details of the costs of new purchases
- When a toy is taken out of use the reason must be recorded so its value can be written off

Task:

1 Write down the data items you think will need to be recorded in order to create this system.

2 Group these items according to subject and use this to create a diagram of the data structure, showing primary keys, foreign keys and relationships.

Use the space on the next page for your design.

Rent-A-Toy Data Structure Diagram

6 ~ Reports

Open the Access database **Students Records 5.accdb**.

Note: You will need to enable active content for some of the exercises.

6.1 - Ungrouped report based on query

We need to produce a printable report that lists all available modules with details of the tutors running them.

1 Create a new query on which to base the report. This needs to include:

- module ID, module name and credits
- Full name and email address of module tutor

2 Save the query as **qryModuleInfo** and check it by viewing the datasheet. (44)

3 Use the report wizard to create a report to list the modules and tutor details, based on this new query. Save the report as **rptModuleInfo**

4 View the report design. Note how the sections of the design are used to generate the pages of the report when viewed in Print Preview.

5 Return to design view and tidy the report, adjusting label and field size/positions.

Note: When viewing a Print Preview, use the **Close Print Preview** button to return to design view.

Extension:

6 Add a field to the report footer to count the total number of modules listed. Use the **Count()** function in an **unbound text box**.

7 Reopen and modify the query to list only modules from the Politics department (deptID POL). After saving and closing the query, view the report again and note the change in the number of modules.

8 Replace the tutor title, forename and surname fields with a single field joining these together.

6.2 - Grouped reports

Use the Access database **Student Records 5.accdb.**

We want to create a report that lists the full names of all first year students and the modules they are taking with the mark they received.

1 Create a new query including the tables and fields you think will be needed.

- Include a filter in the query to only show students currently in year 1. (512)
- Note how this query shows each student multiple times.
- Close and save the query as **qryStudentModules**.

2 Create a grouped report based on this query. Group the report by **studentID** and sort by **modName**.

Applying grouping to the report allows these repeating entries to be appropriately presented. Save the report as **rptStudentModules**.

3 Compare the Design view and Report view and note how the new **studentID** Header section of the design is repeated for each unique student ID.

4 Tidy the report, adjusting label and field size/positions. Depending on how you chose the report grouping you may have some student information in the detail section. If so, move it to the group header.

Extension:

5 Modify the report to include each student's average mark (in an unbound control). You will need to ensure that the footer section is displayed first. This is done through opening the **Group & Sort** window. Modify the properties of this average to display to one decimal place.

6 Add a field in the report footer to show the total number of students. You may need to drag the report footer out from the Design view.

7 Modify the underlying query to only show modules from the Chemistry department (deptID CHE).

7 ~ Forms

Continue using the Access database **Student Records 5.accdb** for these exercises.

7.1 - Form Wizard

1 Create a form using the Form Wizard based on the **Tutors** table. Use all fields except **dept**. Choose the Columnar layout and name the form **frmTutors1**.

2 View the form and use the record selectors to move through the records. Enter a new tutor using the New Record button.

3 Create a second form based on the **Tutors** table, but this time choose a Tabular layout. Name this form **frmTutors2**. Compare this form to frmTutors1.

4 Modify the design of the tabular form to remove the wasted vertical space in some of the fields.

7.2 - Forms based on queries

1 Create a query showing **studentID**, **forename**, **surname**, **email** and **currentYear**. Save the query as **qryStudents.**

2 Use the Form Wizard to create a form based on this query using the columnar layout. Save the form as **frmStudents**. View the form and take a note of the number of records. (500) Close the form.

3 Modify **qryStudents** to only show 1st years. Open **frmStudents**. Note that the same form now only shows 1st years. (170)

7.3 - Subforms

1 Create a new query showing **studentID** and **mark** from the **ModuleChoices** table and **modName**, **credits** and **deptID** from the **Modules** table. Save this as **qryStudentMarks.**

2 Use the Form Wizard to create a form using all fields from this query. Choose to view your data by **ModuleChoices** and select the Tabular layout. Save the form as **frmStudentMarks**.

This form is not very useful as it is, but when used as a subform with the **frmStudents** created earlier it will be very effective.

7.4 - Wizard-free subforms

1 Open **frmStudents** in Design View. Move the form footer down to make more room at the bottom of the form. Use the Controls section of the Design tab to add a sub-form into this space. From the SubForm Wizard, choose to use an existing form and select **frmStudentMarks**. Choose to define your own links and use **studentid** for both the form and the sub-form fields. Name the new subform **frmStudentMarksSubform**.

2 Use the record selectors to cycle through the students. The subform should display the marks that student has attained for each module they have taken. You may need to modify the design so the subform has enough space to display at least 4 results. Save the changes to **frmStudents**.

7.5 - Creating with the wizard

1 Create a query using **roomid**, **college** and **capacity** from the **Rooms** table and **moduleid**, **modName**, **deptID** and **weeks** from the **Modules** table. Save the query as **qryRoomsAndModules**.

2 Use the Form wizard to create a new form based on this query using all fields.

- Choose to view by Rooms. Choose 'Form with subform(s)'.
- Choose the Datasheet layout.

3 Name the form **frmRooms** and the subform **frmRoomsSubform**.

4 Cycle through the records in the subform. Note that not all rooms have modules taken in them.

Extension

5 Modify **frmStudents** to include the average of each student's marks. Use a Text box in the footer of the subform to do this. Set this to display one decimal place.

6 Tidy up the form by removing the record selectors, navigation buttons and scroll bars on the subform and adding more useful title bars.

7.6 - Look-up fields

1 Create a new form based on the **Modules** tables, using **moduleid**, **modName**, **deptID** and **credits**. Choose a columnar layout and save the form as **frmModLookUp**.

2 In Design view, add a combo box control to the form. Choose to get values from another table or query. Select the **Departments** table. Include the **deptid** and **deptname** fields.

- Sort by **deptname**.
- Choose to 'Hide key column'.
- Store the value in **deptID**.

3 Cycle through the records. Note the combo box and the **deptid** fields display corresponding values. Selecting a department name from the combo box changes the value stored in the **deptid** field.