

Section Exercises

The following revision exercises are divided into sections, each targeted at specific elements of the Advanced ECDL syllabus. The individual sections are an exact match for the sections in the ECDL Advanced Training Guides from CiA Training, making the guides an ideal reference source for anyone working through these exercises.

Fundamentals

These revision exercises include topics taken from the following list: database concepts, database development, SQL, creating a table, basic field properties.

Exercise 1

1. What does the term **SQL** stand for?
2. Are the following statements **True** or **False**?
 - a) Access uses SQL to define queries.
 - b) The SQL code behind a query can be displayed in Access.
 - c) You need to know SQL in order to create queries.
 - d) SQL is only used in Access applications.
3. Which two of the following are common types (models) of database?
 - a) Hierarchical.
 - b) SQL.
 - c) Referential.
 - d) Relational.
4. Arrange the following stages of database life cycle into a more realistic sequence:
 - a) Data Entry.
 - b) Database Creation.
 - c) Database Design.
 - d) Information Retrieval.
 - e) Data Maintenance

5. A manufacturing company has a single integrated database system which holds data for many of its functions, including manufacturing, purchasing, accounting and personnel. Is this an example of a ?
 - a) Website content management system.
 - b) Enterprise resource planning system.
 - c) Customer relationship management system.
 - d) Dynamic website.

6. Which type of object must be present in an Access database: table, query or form?

Exercise 2

1. Which of the following database operations would require a technical knowledge of *Microsoft Access*?
 - a) Add data to a table.
 - b) Add a new field to a table.
 - c) Change the columns in a report.
 - d) Run a report.

2. Create a new database to log invoices for a scientific supplies company. Name the database **Invoices**.

3. Create the following table to be called **Details**. Select appropriate data types and field sizes for each field and add the first four records as shown below.

| Invoice | Due | Customer Name | Customer Address | Item | Quantity | Price | Total Value |
|---------|-----------------|---------------------|------------------|---------------|----------|-----------|-------------|
| 10012 | 05 January 2009 | Conrads | 27 High Street | Neutron Tubes | 6 | £1,500.00 | £9,000.00 |
| 10016 | 06 January 2009 | HiTek | Century Building | Spectrometer | 1 | £3,000.00 | £3,000.00 |
| 10022 | 11 January 2009 | Conrads | 27 High Street | Power Packs | 5 | £75.00 | £375.00 |
| 10029 | 12 January 2009 | The Johnson Company | Johnson House | Power Packs | 10 | £75.00 | £750.00 |

4. Give any reasons why this is not the most efficient design for a database.

Revision Series

5. Replace the **Customer Name** and **Customer Address** fields with a single **Customer Reference** field (length 6). Use the following information to enter the customer information.

| Customer Reference | Customer Name |
|---------------------------|----------------------|
| C001 | Conrads |
| H001 | HiTek |
| J001 | The Johnson Company |

6. Remove the **Total Value** field from the table.
7. Change the format of the **Price** field to show **Euros**.
8. Change the format of the **Due** field to **Long Date**.
9. Adjust column widths so that all the headings and data are fully displayed.
10. Close the table, saving the design, then close the database.

Field Properties

These revision exercises include topics taken from the following list: creating lookup fields, changing formatting options, setting and modifying default values, setting mandatory fields, creating validation rules and text, creating input masks.

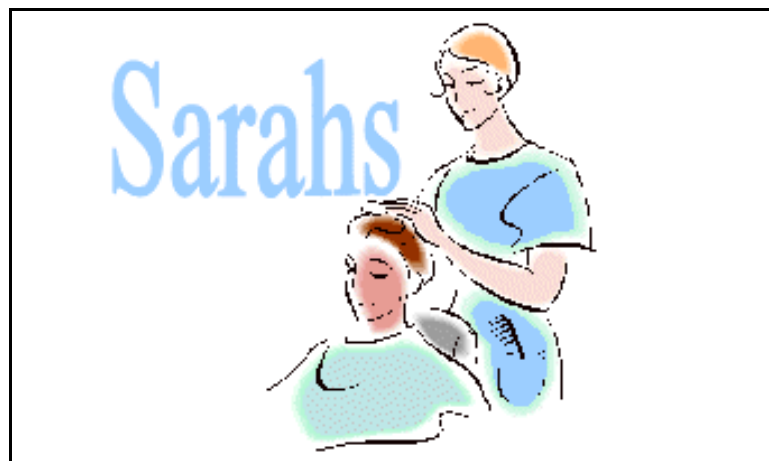
Exercise 3

1. Open the **Sarabs Salon** database. This has a table of **Bookings** and related tables for **Processes**, **Rooms** and **Technicians**. Look at the data in all the tables then open the **Bookings** table in **Design View**.
2. Change the **Field Properties** for **Date** so that it defaults to today's date (with no time component). The default value must change as the current date changes.
3. Create an input mask for the **Time** field based on the **Short Time** format. What is the resulting mask?

4. Create a validation rule for **Time** so that times greater than **17:30** will not be accepted. The validation text is to be **No bookings after 5:30 pm**.
5. Create a **Lookup** field for **Technician Name** that looks up the **Name** field from the **Technician** table. Restrict entries to those on the list.
6. Create an input mask for the **Room No** field so that only one character and one number will be accepted. The character will be converted automatically to upper case. What is the resulting mask?
7. Switch to **Datasheet View** and create a new record by typing in the following values (date should already be present as today's date):

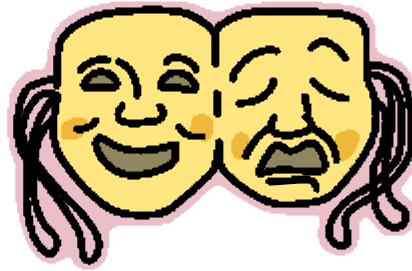
| Ref | Name | Date | Time | Process | Technician | Room | Comment |
|------|-----------|---------|------|----------|------------|------|---------|
| 1409 | June Ball | 'today' | 1800 | Pedicure | Bronwyn | aa22 | Test |

8. You will be halted at the **Time** field (amend the entry to **1700**) and the **Technician** field (amend to **Lucy**). Confirm that because of the input masks, **1700** will appear as **17:00** and **aa22** will appear as **A2**.
9. Even though the field properties validations are all met, the new record still cannot be added. Why is this?
10. Change the **Process** to **Manicure** and add the record.
11. Add **Bronwyn** to the **Technician Name** table then amend the new booking record to show that name.
12. Close the database.



Exercise 4

1. Create a new database called **Booknow** to record telephone bookings for a theatre from registered customers.
2. Create a table called **Details** with the fields **Customer ID** (Text), **When** (Date/Time), **Performance** (Text), **Location** (Text), **No of Tickets** (Number). Do not define a primary key.



3. **Customer ID** values are always two upper case characters followed by 3 digits, e.g. **AB123**. Define an input mask to ensure that this format is always applied.
4. The **Performance** field can only be **Matinee**, **Early** or **Late**. Define a validation rule so that only one of these values can be entered.
5. The **Location** field can only be **Front Stalls**, **Rear Stalls**, or **Circle**. Define this field as a **Lookup** field (from a typed list) so that only one of these values can be selected.
6. What are the advantages of using a **Lookup** field rather than a **Validation Rule** to restrict entries to a limited set of values?
7. Add **Upper Circle** to the list of acceptable locations.
8. No more than three tickets can be booked on any one **Customer ID**. Set the maximum acceptable **No of Tickets** to **3** and define a suitable message to be displayed if the condition is not met.
9. A booking cannot be made unless all fields are completed. Make every field in the table mandatory.
10. Close the table, saving any changes if prompted and close the database.

Queries

These revision exercises include topics taken from the following list: using wildcards in queries, creating summary queries showing sum, count, average, maximum and minimum values, creating calculated fields using arithmetic operations, creating two variable parameter queries.

Exercise 5

1. Open the database **Periodic** showing details of some of the elements as classified in the Periodic table. Open the **Elements** table and examine the data.
2. Create and save a query called **Query1** to list the **Atomic Number**, **Name** and **Symbol**. Use a single wildcard criteria to select only elements whose symbol is the letter "c" followed by any vowel. What is the wildcard criteria and how many elements are selected?
3. Create and save a query called **Query2** to list the **Atomic Number**, **Name**, **Symbol** and **Boiling Point** for all elements whose boiling point is between **0°C** and **100°C**. How many are there?
4. The temperatures in the table are shown in degrees centigrade. It is also possible to show temperatures as degrees above absolute zero, or degrees absolute (sometimes called degrees Kelvin), where absolute zero is minus 273 degrees centigrade. Create a query to list the **Atomic Number**, **Name**, and **Symbol** for all elements classified as gases. Do not display the **Classification** field. Add a calculated field **Absolute**, which is defined as the **Boiling Point + 273**. How many degrees above absolute zero does liquid helium boil? Save the query as **Query4**.
5. Create a query called **Query5** to group the elements by **Classification** and show a count of the element names under each classification heading. How many are classified as liquids?
6. On the same query, add columns to show the average melting point and average boiling point for each classification.

7. In **Design View**, amend the field properties for the two average columns so that the data is displayed with a format of **Fixed**.
8. Create and save a query called **Query6** to list the **Atomic Number**, **Name** and **Symbol** for all elements not classified as metal. How many are there?
9. Make sure all queries are saved then close the database.

Exercise 6

1. Open the database **Commercial** showing details of some commercial premises for sale. Open the **Premises** table and examine the data.
2. Create and save a query called **Analysis1** to group the data by **Type of Premises** and show the average premises price in each category. Make sure the average price field is formatted as **Currency**. What type of premises has the lowest average price?
3. Create and save a query called **Analysis2** to group the data by **Location** and show the average price, the maximum price and the minimum price for each area. Make sure all fields are formatted as **Currency**.
4. Sort the query so that the location with the highest average price is shown first. Which location has the lowest average price?
5. Create and save a query called **Query1** to list the **Premises ID**, **Location**, **Address** and **Type of Premises**.
6. Add a calculated field called **Rate** to show the price per unit area for each premises. Format the new field as currency.
7. Change the query so that only premises with a rate value less than **500** and that are not **Manufacturing Units** are displayed. How many are there?
8. Save the query but leave it open.
9. Remove the **Type of Premises** selection criteria and replace the calculated field with the **Price** field.

10. Sort the data in ascending order of **Price** and use a setting on the toolbar/ribbon to display only the top three records in the sort order. Save the query as **Query2** and leave it open.
11. Remove the 'top three' selection and include a selection criteria so that the query will prompt for a **Type of Premises** value and an upper limit for **Price**, and only display the relevant records. Enter the text for the first prompt as **Which type?** and the second as **Up to Price?**
12. Run the query for **Store Unit** premises, up to **£80,000**. How many records are selected?
13. Save the query as **Query3** then close it and close the database.

Relationships

These revision exercises include topics taken from the following list: applying primary keys, applying and modifying different types of relationship (one to one, one to many, many to many, understanding joins (inner, outer, subtract, self), applying referential integrity and cascade options.

Exercise 7

1. Open the **Golf** database showing a table of senior members and their playing partners for a forthcoming pairs competition.
2. Create a query which will list all fields from the **Pairings** table, but also show the **Handicap** for the second person in the pairing. This will need a self join in the query, from the **Pairings** table to a copy of it.
3. Save the query as **Query1** and close the database.
4. Open the database **Salon** showing part of the advanced bookings data for a beauty salon. There are tables for **Bookings**, **Processes** and **Rooms**, but no relationships between them.
5. Apply primary keys to the first fields in the **Processes** and **Rooms** tables so that they can be used in **One to Many** relationships with the **Bookings** table.

6. Create **One to Many** relationships from both the **Processes** and **Rooms** to the **Bookings** table. Apply **Referential Integrity** to both relationships and select all cascade options.
7. Print out the relationships diagram.
8. Create a query showing **Booking Ref**, **Customer Name** and **Date** from the **Bookings** table with **Description**, **Duration** and **Charge** from the **Processes** table. Save the query as **Billing**.
9. View the **Rooms** table in **Datasheet View** and expand the subdatasheets to find which room is double booked (bookings on the same time and date). What are the details of the double booking?
10. Create a summary query based on the **Rooms** and **Bookings** tables, which counts the number of bookings, for each **Room Number**. Save the query as **Usage**. Why is there no record for room **A4**?
11. Change the join type for the **Rooms - Booking** link so that data for all rooms, including those with no usage, is shown in the query. What is the new type of join called?
12. Save the query and close the database.

Exercise 8

1. Open the database **Letting** showing part of the booking register for some holiday apartments. There are tables for **Units**, describing each apartment, and **Bookings**, listing the details of bookings for the apartments. There are no relationships between them.



2. Attempt to create a link with **Referential Integrity** between **Units** and **Bookings** based on the **Apartment** field. Why is this not allowed?
3. Cancel the relationship process without saving. Correct the problem by changing the design of the **Bookings** table, then create the link as described above. Do not select the cascade options.
4. Create a summary query to shows the total number nights from the **Booking** table for each **Apartment** on the **Units** table. Save the query as **Totals**.
5. Change the query to list only apartments that have no bookings. This will involve changing the **Join Type** to a **Subtract** join by using a selection. Save the amended query as **None**.
6. Apartment **35** is no longer available due to storm damage. Open the **Units** table and attempt to delete the record for apartment **35**. What is the outcome?
7. Edit the relationship between the tables so the cascade options are selected. Try to delete the record for **apartment 35** again. What is the outcome now?
8. Complete the deletion of the record and close the database.

Forms

These revision exercises include topics taken from the following list: creating main forms and sub forms, using subform wizard, linking forms.

Exercise 9

1. Open the **Sunny Holidays** database.
2. Use a wizard to create a form based on the **Units** table with a subform based on the **Bookings** table. Name the forms



Units2 and **Bookings2**. **Units2** should contain the fields **Unit**, **Location**, **Type**, **Beds** and **Weekly Charge**. **Bookings2** should contain the fields **Start Date**, **Nights** and **Name**. Specify a **Tabular** layout and a style of your choice.

Revision Series

3. When the form has been created add a new control to the right of **Weekly Charge** which calculates **5%** of the **Weekly Charge** value. Format the control as **Currency** and add a caption of **Early Payment Discount**.
4. Switch to **Form View** and add a new **Booking** in the subform for unit **S17** using your name.
5. Print a copy of the form showing this record.
6. Switch to **Design View** and delete the whole subform.
7. Make sure **Control Wizards** feature is switched on and use a button to add a new subform in the same position as before. The subform is to be based on the **Bookings** table and include the fields **Start Date**, **Nights** and **Name**. Show bookings for each unit, and call the subform **Bookings3**.
8. Use the new form to locate the record added in step 4.
9. Print a copy of the form showing this record then delete the record.
10. Close the form and close the database.

Exercise 10

1. Open the database **Saraha's Salon** and create a new query in **Design View**, based on the **Processes** and **Bookings** tables.
2. Add the following fields to the query grid in the order shown:
The **Description** field from the **Processes** table.
Booking Ref, **Customer Name**, **Date**, **Time** and **Room No** from the **Bookings** table.
Save the query as **Details**.
3. Use a wizard to create a form and subform based on every field from the **Details** query. The data should be viewed by **Processes**. Specify a **Datasheet** layout and any style. Name the form **Booking Details**, and name the subform **Booking Subform**.

4. Use a wizard to create a form based on the **Bookings** table including the fields **Booking Ref**, **Customer Name**, **Date**, **Time** and **Room No**. Specify a **Datasheet** layout and any style. Give the form a title of **Manual Subform**.
5. Create a form in **Design View** including all the fields from the **Processes** table.
6. Without using the wizard, insert a subform based on the **Manual Subform** form below the existing fields. Use **Process** as the field to link the main form to the subform and give the form a title of **Manual Details**.
7. Enlarge the size of the subform if necessary to display all fields fully.
8. Locate the record for **Manicure** in the **Manual Details** form. How many bookings exist for this process?
9. Close the forms and the database.