DATABASES - LAB

2020-2021

الدراسات الاولية / البكالوريوس / الفصل الثاني

الكادر التدريسي

م.د مهدي دعيمي كزار

م.د سرمد مكي محمد غريب

م.م. فراس محمد كاظم

ر. مبرمجين اقدم بيداء سلمان قنديل

2020-
2021

Databases - Lab

Third Class – Second Semester

Morning & Evening - Undergraduate Studies

قواعد البيانات

المرحلة الثالثة – الفصل الدراسي الثاني – (العملي)

الدراسات الاولية – الدراسة الصباحية & المسائية

Assist. Lecturer: Firas M. Kadhum

fmk@sc.uobaghdad.edu.iq

www.sites.google.com/site/firasmkadhum

2- Queries:

- ✓ As previously mentioned, you used a **MS Access Query** for defining the structure (schema) of the database (by using **Data Definition Queries**).
- ✓ Now, you can use a **MS Access Query** as a **question (selecting)** that you ask about the information stored in a database. And also, you can use it for **inserting**, **deleting** and **updating** data in a database.
- ✓ **MS Access** supports many different **Types of Queries**, as follows:
  - **Select Query**
  - **Total Query**
  - **Crosstab Query**
  - **Pass-Through Query (SQL Specific Query)**
  - **Union Query (SQL Specific Query)**
  - **Join Query**
  - **Action Query (Make-Table, Append, Update or Delete)**
  - **Subquery, etc...**

  We'll discuss each type, in the next section and later.

❖Select Query:

- ✓ The most **common** type of query is the **Select Query**.
- ✓ As its name implies, a **Select Query** selects information from **one** or **more Tables/Queries**, creating a **recordset** (**Query's result**).

> **Tip**: Remember that, unlike tables, **recordsets** don't physically exist in your database. Instead, MS Access stores the query, and it only displays a **recordset** when you run the query.

- ✓ MS Access provides three **Ways** to create **Select Query** using;
  - the **SQL View**,
  - the **Query Wizard**,
  - and the **Query Design**.

1- Using SQL View:

Whether you are familiar with **SQL** (**S**tructured **Q**uery **L**anguage) or not, you can write, edit or view your query using **SQL View**.

Tip: To switch to **SQL View**;

1- Open the query in **Design** or **Datasheet View**.

2- Then click on the drop-down **View** list and then select the **SQL > SQL View**.

3- When you finish writing or changing the SQL statements as needed, go to the **Design** tab, and in the **Results** group, click **Run**. Your **recordset** appears as a **datasheet**

SELECT Statement:

Syntax:

SELECT [*predicate*] { * | *table*.* | [*table*.]*field1* [**AS** *alias1*] [, [*table*.]*field2* [**AS** *alias2*] [, ...]]}

FROM *tableexpression* [, ...] [**IN** *externaldatabase*]

[WHERE ...]

[GROUP BY ...]

[HAVING ...]

[ORDER BY ...]

The **SELECT** statement has these parts:

Part	Description
predicate	One of the following predicates: ALL, DISTINCT, DISTINCTROW, or TOP. You use the predicate to restrict the number of records returned. If none is specified, the default is ALL.
*	Specifies that all fields from the specified table or tables are selected.
table	The name of the table containing the fields from which records are selected.
field1, field2	The names of the fields containing the data you want to retrieve. If you include more than one field, they are retrieved in the order listed.
alias1, alias2	The names to use as column headers instead of the original column names in *table*.
tableexpression	The name of the table or tables containing the data you want to retrieve.
externaldatabase	The name of the database containing the tables in *tableexpression* if they are not in the current database.

Notes: 1- **SELECT** statements do not change data in the database.

2- The minimum **syntax** for a **SELECT** statement is: **SELECT** ***fields*** **FROM** ***table***

3- You can use an asterisk (*) to select all fields in a table.

2- Using Query Wizard:

The **Query Wizard** is the **easiest** way to get started building queries, especially if you're new to MS Access.

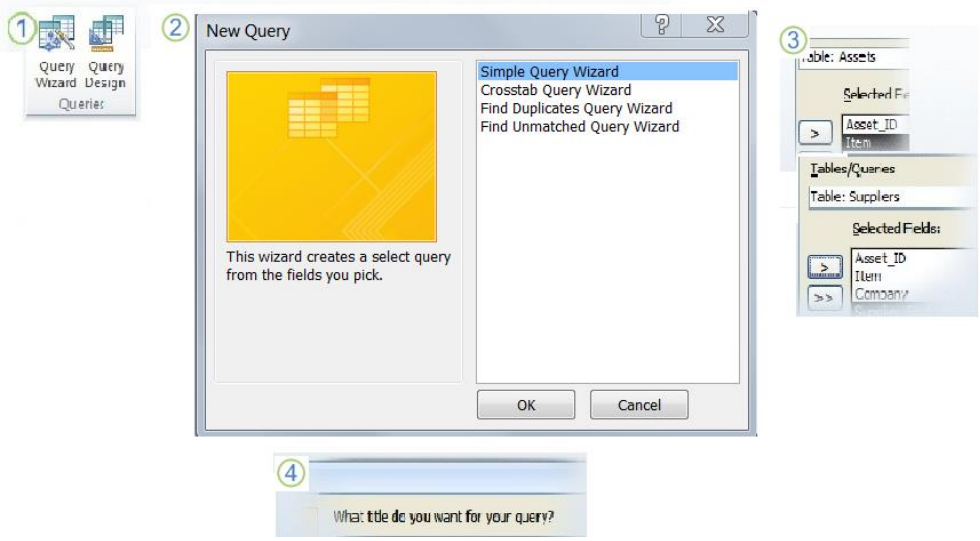


Figure 1-1: The Query Wizard process

1. On the **Create** tab, in the **Queries** group, click **Query Wizard**.
2. After clicked on **Query Wizard**, the **New Query** dialog box is displayed, which provides **four** basic types of generic queries as shown in Figure 1-1.

1	**Simple Query Wizard**	Create a select query from the fields you select.
2	**Crosstab Query Wizard**	Summaries the query data in spreadsheet formats.
3	**Find Duplicates Query Wizard**	Locates duplicate records in a query.
4	**Find Unmatched Query Wizard**	Locates records in one table that do not have matching records in a related table.

3. Complete the wizard. As part of that process, select the **tables/queries** and **fields** you want to use as your record source, and…
4. Give your new query a descriptive name, and remember to avoid using spaces in the name.

3- Using Query Design:

✓ The **Query Design** consists of **two** sections:

1- The table/query pane: This is where tables or queries and their fields are added to the query's design.

2- The Query By Example (QBE) grid: The **QBE** grid holds the field names involved in the query and any criteria used to select records. Each column in the **QBE** grid contains information about a single field from a table or query contained within the upper pane.

The **QBE** grid has **six** labeled rows:

1	**Field**	This is where **field names** are entered or added.
2	**Table**	This row shows the **table** the field is from. This is useful in queries with multiple tables.
3	**Sort**	This row enables **sorting** instructions for the query.
4	**Show**	This row determines whether to **display** the field in the returned recordset.
5	**Criteria**	This row consists of the **criteria** that filter the returned records, you can see some examples of criteria in Table 1-2.
6	**or**	This row is the first of a number of rows to which **you can add multiple query criteria**.

- ✓ When working with **Select** queries, you may need to specify **one** or **more criteria** to limit the scope of information returned. You specify criteria by using some **operators**.
- ✓ In **Select** queries, **operators** are used in either the **Field** or **Criteria** cell of the **QBE** pane. Table 1.1 shows the most common operators used in **Select** queries.

Table 1.1: Common Operators Used in Select Queries

Mathematical	Relational	Logical	String	Miscellaneous
* (multiply)	= (equal)	And	& (concatenate)	Between...And
/ (divide)	<> (not equal)	Or	Like	In
+ (add)	> (greater than)	Not	Not Like	Is Null
– (subtract)	< (less than)			Is Not Null

Note: The **Query Window** has **three Views**:

1) **Design View**: Design view is where you create the query.
2) **Datasheet View**: Datasheet view displays the records returned by the query.
3) **SQL View**: The SQL View window displays the SQL statement behind a query.

CRITERIA	EFFECT
> 234	Returns all numbers greater than 234. To find all numbers less than 234, use < 234.
>= "Callahan"	Returns all records from Callahan through the end of the alphabet.
Between #2/2/2007# And #12/1/2007#	Returns dates from 2-Feb-07 through 1-Dec-07 (ANSI-89). If your database uses the ANSI-92 wildcard characters, use single quotation marks (') instead of pound signs. Example: Between '2/2/2007' And '12/1/2007'.
Not "Germany"	Finds all records where the exact contents of the field are not exactly equal to "Germany." The criterion will return records that contain characters in addition to "Germany," such as "Germany (euro)" or "Europe (Germany)".
Not "T*"	Finds all records except those beginning with T. If your database uses the ANSI-92 wildcard character set, use the percent sign (%) instead of the asterisk (*).
Not "*t"	Finds all records that do not end with t. If your database uses the ANSI-92 wildcard character set, use the percent sign instead of the asterisk.
In(Canada,UK)	In a list, finds all records containing Canada or UK.
Like "[A-D]*"	In a Text field, finds all records that start with the letters A through D. If your database uses the ANSI-92 wildcard character set, use the percent sign instead of the asterisk.
Like "*ar*"	Finds all records that include the letter sequence "ar". If your database uses the ANSI-92 wildcard character set, use the percent sign instead of the asterisk.
Like "Maison Dewe?"	Finds all records that begin with "Maison" and that also contain a 5-letter second string in which the first 4 letters are "Dewe" and the last letter is unknown (indicated by a question mark). If your database uses the ANSI-92 wildcard character set, use the underscore (_) instead of the question mark.
#2/2/2007#	Finds all records for February 2, 2007. If your database uses the ANSI-92 wildcard character set, surround the date with single quotation marks instead of pound signs (#). Example: '2/2/2007'.
< Date() - 30	Returns all dates more than 30 days old.
Date()	Returns all records containing today's date.
Between Date() And DateAdd("M", 3, Date())	Returns all records between today's date and three months from today's date.
Is Null	Returns all records that contain a null (blank or undefined) value.
Is Not Null	Returns all records that contain a value.

Table 1-2: Some examples of criteria

Here's the process in Query Design;

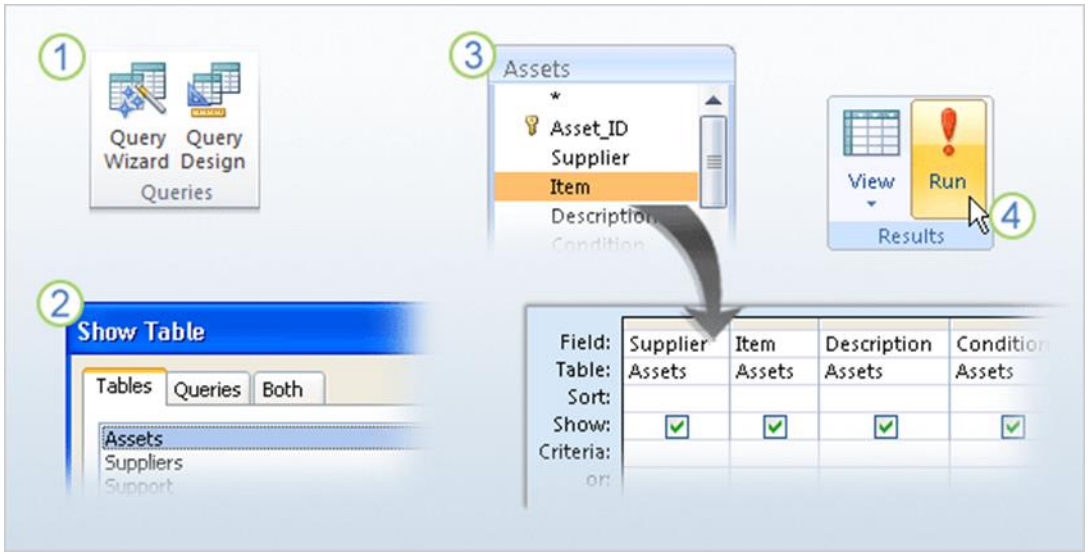


Figure 1-2: The Query Design process

Assignment (II-1)

Using **Suppliers-Parts-Shipments** database (**Suppliers-Parts-Shipments.accdb**) that you created in the previous laboratory, do the following queries:

1) Get full details of all parts?

2) Retrieve the part ID (PID) for all parts supplied?

3) Get the part ID (PID) for all parts supplied without duplicate?

4) For all parts, gets the part ID (PID) and weight (WEIGHT) of that part in grams (where the part weight are given in Pounds; Grams = Pounds * 454)?

5) List the supplier name (SNAME) that the second letter is L?

6) Get the supplier ID (SID) and status (STATUS) for suppliers in Paris?

7) Retrieve the supplier ID (SID) and Name (SNAME) for suppliers in Paris with status (STATUS) more than 20?

8) Get full details of parts that are not Blue or Red?

9) Get the supplier ID (SID) and status (STATUS) for suppliers in Paris in descending order of status (STATUS)?

10) Sort the SP (SHIPMENTS) table according to the supplier ID (SID) ascending and the part ID (PID) descending?

❖ Total Query:

✓ A **Total Query** is a special type of **Select Query**. **Total** queries provide **Sum** or other calculations (such as **Count, Avg, Min, Max etc.**) from the records returned by a **Select Query**.

✓ MS Access performs these calculations by using **nine** aggregate functions that let you determine a specific value based on the contents of a field. For example, you can determine;

- the average price for products by type,
- the maximum and minimum price paid for a product,
- or the total count of all products purchased by customers in particular states.

Performing each of these examples as a query results in a **recordset** based on the calculations you requested.

✓ MS Access provides three **Ways** to create **Total Query** using;

- the **SQL View,**
- the **Query Wizard,**
- and the **Query Design**.

1- Using SQL View: (see previous lab)

2- Using Query Wizard: (see previous lab)

3- Using Query Design: to create a **Total Query**, you use a new row in the **Query By Example (QBE)** pane — the **Total** row. The following section describes this handy tool in detail.

- **Showing and hiding the Total row in the QBE pane:**

1) To create a **totals query**, create a select query and then display the **Total** row of the **QBE** pane.

2) You open the **Total** row by clicking on the **Totals** button in the **Show/Hide** group on the **Design** ribbon tab while a query is open in the **Query Designer**.

Tip: If the **Table** row is not present on your screen, open it by clicking the **Table Names** button in the **Show/Hide** group on the **Design** ribbon tab.

The options in Total row:

- You can perform total calculations on **all records** or **groups of records** in **one** or **more Tables/Queries**.
- To perform a calculation, you must select one of the options from the drop-down list in the **Total** row for every field you include in the query, including any hidden fields (with the **Show** option turned off). Figure 2.1 shows the drop-down list box opened in the **Total** row of the ***QTY*** field in ***SP*** table.

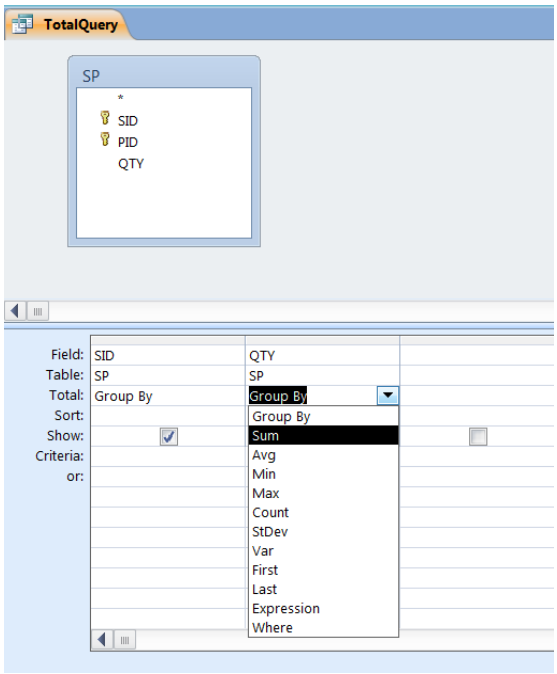


Figure 2-1: The options in Total row

Tip: What is an aggregate function?
An **aggregate function** takes a group of records and performs a mathematical/statistical operation over the entire group.

✓ The **options** can be divided into **four** categories: **Group By**, **Aggregate Function**, **Expression**, and **Where**. Table 2.1, lists each category and its purpose.

Four Types of Total Options

Category	Purpose of Operator
Group By	Groups common records together. Access performs aggregate calculations against the groups.
Aggregate	Specifies a mathematical or selection operation to perform against a field. There are nine different aggregate functions in Access 2010.
	Sum - totals the field values. **Avg** - computes an average field value. **Min** - finds the minimum field value in a field. **Max** - finds the maximum filed value in a field. **Count** - returns the number of values in a field, **disregarding NULL (unknown) values**. **StDev** - computes the Standard deviation of values in a field. **Var** - computes the Variance of values in a field. **First** - returns the first value in a field. **Last** - returns the last value in a field.
Expression	Enable you to add expression or to create a calculated field for a group.
Where	Filters records **before** performing an aggregate calculation against a field.

Assignment (II-2)

Using **Suppliers-Parts-Shipments** database (**Suppliers-Parts-Shipments.accdb**) that you created in the previous laboratory, do the following queries:

1) Retrieve the overall summation of shipped quantities?

2) Get the total number of suppliers?

3) Retrieve the total number of suppliers, currently supplying part(s)?

4) Get the total number of parts that have stored in 'London' city?

5) Get the summation of quantities for supplied part 'P1'.

6) Retrieve the supplier ID and max shipment quantity, for each supplier?

7) For each shipped part, get the part ID and the total quantity for that part?

8) For each supplied part, get the part ID and the total quantity for that part, except those with a sum < 375?

9) Get the PID for all parts supplied by more than **one** supplier?

10) Get the number of shipments supplied by 'Jones'?

11) Get the part ID and the total quantity that exceed 450 for each part, excluding the quantity = 100 and sort the result according to part ID descending?

❖ Crosstab Query:

✓ MS Access supports a specialized type of **Total Query** — the **Crosstab Query** — that summarizes data in a **row**-*and*-**column** format.

✓ So, a **Crosstab Query** is a **spreadsheet-like** summary of the things specified by the row and column headers created from your tables.

✓ MS Access provides three **Ways** to create **Crosstab Query** using;

- the **SQL View**,
- the **Query Wizard**,
- and the **Query Design**.

1- Using SQL View:

TRANSFORM Statement:

Syntax:

TRANSFORM ***aggfunction***

selectstatement

PIVOT ***pivotfield*** **[IN (*value1* [, *value2* [, ...]])]**

The **TRANSFORM** statement has these parts:

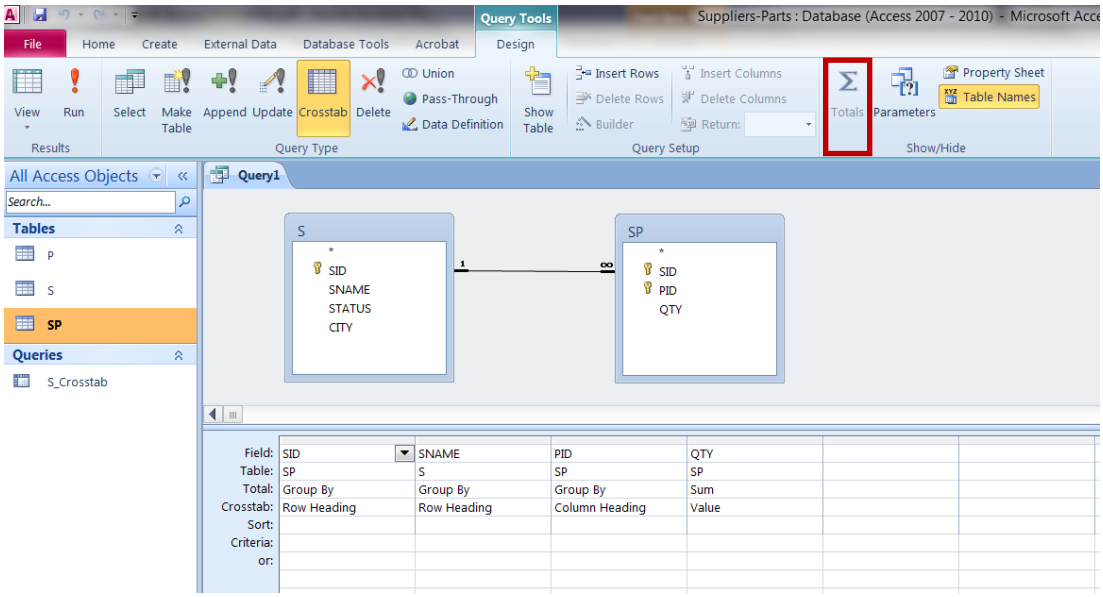
Part	Description
aggfunction	An SQL aggregate function that operates on the selected data.
selectstatement	A SELECT statement.
pivotfield	The field or expression you want to use to create column headings in the query's result set.
value1, value2	Fixed values used to create column headings.

Tip: Optionally, you can include other clauses, such as **WHERE**, that specify additional selection. You can also use **Subqueries** — specifically, those in the **WHERE** clause — in a crosstab query.

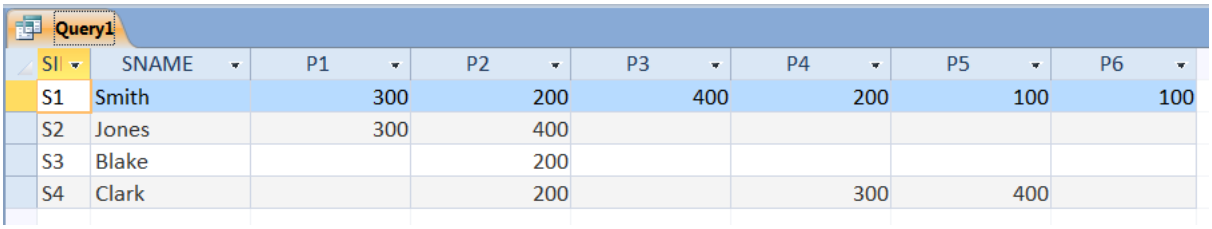
2- Using Query Wizard: (Crosstab Query Wizard)

3- Using Query Design:

to create a **Crosstab Query**, you use a new row **(Crosstab row)** in the **Query By Example (QBE)** pane. In this specialized type of **Total Query**, the **Total row** in the **QBE** pane is always active and cannot be toggled off in a **Crosstab Query**, as depicted in figure 3-1.



The result:



Query1

SI	SNAME	P1	P2	P3	P4	P5	P6
S1	Smith	300	200	400	200	100	100
S2	Jones	300	400				
S3	Blake		200				
S4	Clark		200		300	400	

Figure 3-1: Crosstab Query

Notes:

1- The easiest way to construct a **Crosstab Query** is with the **Crosstab Query Wizard**. The **Crosstab Query Wizard** is an excellent tool to help you create a simple crosstab query quickly.

2- The fields used as **rows** and **columns** must always have **Group By** in the **Total row**. **Otherwise**, Access reports an error when you attempt to display or run the query.

❖ PARAMETER Query: (Pass-through Query)

✓ For queries that you run regularly, you can use a **PARAMETERS** declaration to create a **Parameter Query**.

✓ A **Parameter Query** can help automate the process of changing query criteria.

✓ With a **Parameter Query**, your code will need to provide the parameters each time the query is run.

PARAMETERS Declaration:

Syntax:

PARAMETERS ***name1 datatype1*** **[,** ***name2 datatype2*** **[, ...]] ;**

The **PARAMETERS** declaration has these parts:

Part	Description
name	The name of the parameter. Assigned to the **Name** property of the **Parameter** object and used to identify this parameter in the **Parameters** collection. You can use *name* as a string that is displayed in a dialog box while your application runs the query. Use brackets ([ ]) to enclose text that contains spaces or punctuation. For example, [Low price] and [Begin report with which month?] are valid *name* arguments.
datatype	One of the primary Microsoft Access SQL data types or their synonyms.

Microsoft Access data type	SQL Data Types
Yes/No	BIT
AutoNumber	COUNTER
CURRENCY	CURRENCY
DATE/TIME	DATETIME
MEMO	LONGTEXT
NUMBER (FieldSize= SINGLE)	SINGLE
NUMBER (FieldSize= DOUBLE)	DOUBLE
NUMBER (FieldSize= BYTE)	BYTE
NUMBER (FieldSize= INTEGER)	SHORT
NUMBER (FieldSize= LONG INTEGER)	LONG
OLE	LONGBINARY
TEXT	TEXT

Table 3-1: Mapping Data Types

Notes:

1- The **PARAMETERS** declaration is **optional** but when included **precedes** any other statements.

2- If the declaration includes **more than one** parameter, separate them with **commas**. The following **example** includes **two** parameters:

```
PARAMETERS Low_Price Currency, Beginning_Date DateTime;
```

3- You can use **name** but not **datatype** in a **Where** or **Having** clause. The following example expects **two** parameters to be provided and then applies the criteria to records in the **Orders table**:

```
PARAMETERS Low_Price Currency, Beginning_Date DateTime;
SELECT OrderID, OrderAmount
FROM Orders
WHERE OrderAmount > Low_Price AND OrderDate >= Beginning_Date;
```

Assignment (II-3)

Using **Suppliers-Parts-Shipments** database (**Suppliers-Parts-Shipments.accdb**) that you created in the previous laboratory, do the following queries:

1) Make the following table using Crosstab Query :

SID	P1	P2	P4	P6
S1	1	1	1	1
S2	1	1		
S3		1		
S4		1	1	

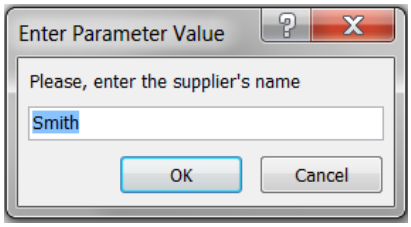
2) Get the following result:

SID	Total	P1	P2	P3	P4	P5	P6
S1	6	300	200	400	200	100	100
S2	2	300	400				
S4	3		200		300	400	

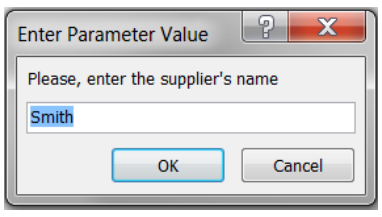
3) Make the following table using Crosstab Query:

SNAME	The Sum of Supplied Parts	Blot	Cam	Cog	Nut	Screw
Blake	200	200				
Clark	900	200	400			300
Jones	700	400			300	
Smith	1300	200	100	100	300	600

4) Retrieve all information of a specific supplier depending on his/her name that you entered it at run-time, give the name of parameter as [Please, enter the supplier's name].



5) Do the same **Q3**, but for each supplier by giving his/her name at run-time, for example:



Result

SNAME	The Sum of Supplied Parts	Blot	Cam	Cog	Nut	Screw
Smith	1300	200	100	100	300	600

6) Get full details of shipments that have quantities between two values (range) during run-time, give the name of parameters as [Please, enter the First Qty] and [Please, enter the Last Qty].

7) Get full details of shipments that have quantities between [Please, enter the First Qty] and [Please, enter the Last Qty], during run-time?

8) List all statuses of suppliers which are multiplied by a value will be given during the execution, use [Please, enter the value :] as name of parameter?

9) Get the number of shipments supplied for a specific supplier by using his/her name that you entered it at run-time, give the name of parameter as [Please, enter the supplier's name].

10) For each part, get the part ID and the total quantity that exceed a value will be given during run-time, use [Please, enter the value :] as name of parameter, excluding the quantity = 100 and sort the result according to part ID descending?

❖Join Query:

There are different types of SQL joins, see Figure 4-1. In the following sections, you learn about a number of different types of join query:

- **Cartesian product (cross-product)**
- **Inner join (equi-join)**
- **Left | Right join**
- **Self-join**

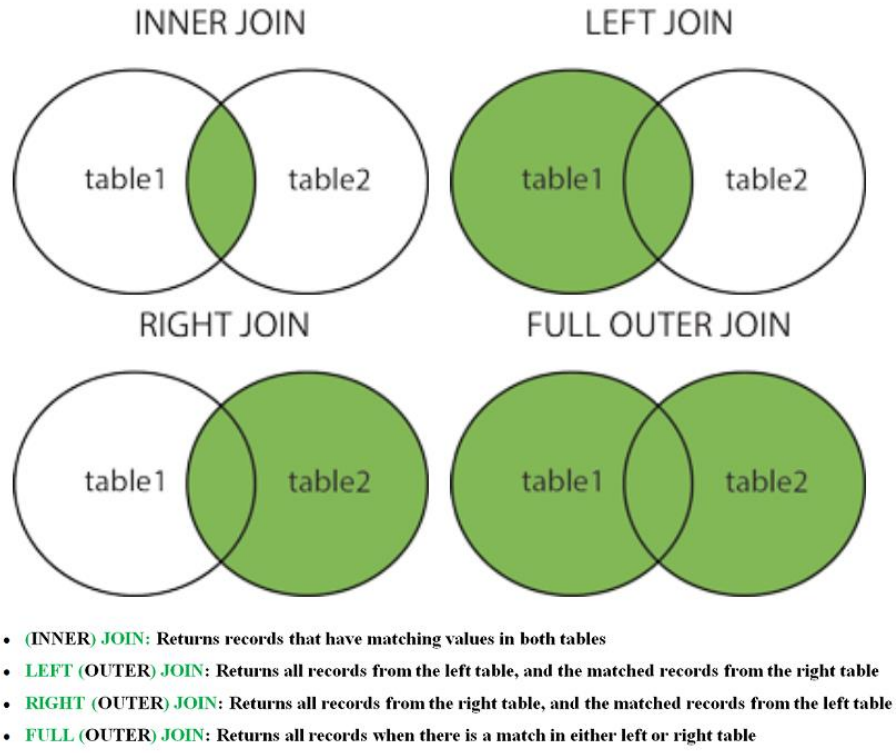


Figure 4-1: Types of SQL Joins

Cartesian product:

Combining all records in one table with all record in another table results in a Cartesian product (**Cross-product**) of the tables.

Inner join (equi-join):

- ✓ The **INNER JOIN** as depicted in Figure 4-2, also known as an **equi-join**, is the most commonly used type of join.
- ✓ This join is used to retrieve rows from **two** or **more** tables by matching a field value that is common between the tables.
- ✓ The fields you join on must have similar **data types**, and you cannot join on Memo or OLE data types.

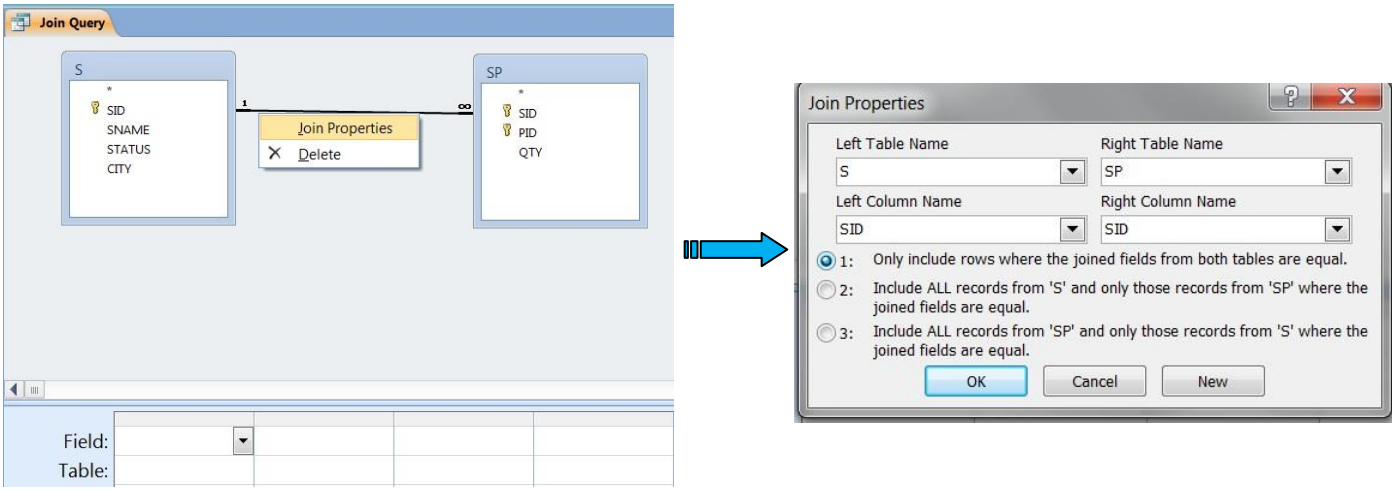


Figure 4-2: Join Properties

Syntax:

```
SELECT ...
FROM table1 INNER JOIN table2
ON table1.field1 Comp_opr table2.field2
```

The **INNER JOIN** has these parts:

Part	Description
table1, *table2*	The names of the tables from which records are combined.
field1, *field2*	The names of the fields that are joined. The fields must be of the same data type and contain the same kind of data, but they do not need to have the same name.
Comp_opr	Any relational comparison operator: "=," "<," ">," "<=," ">=," or "<>."

Left | Right join:

- ✓ Unlike **INNER JOIN**, **Left** | **Right** join shows all records in one table and any matching records in the other.
- ✓ The table or query that does not have a matching record simply displays an **empty cell** for the unmatched data when the recordset is displayed.
- ✓ A **LEFT JOIN** or a **RIGHT JOIN** may be nested inside an **INNER JOIN**, but an **INNER JOIN** may not be nested inside a **LEFT JOIN** or a **RIGHT JOIN**.

Syntax:

```
SELECT ...
FROM table1 { LEFT | RIGHT } JOIN table2
ON table1.field1 Comp_opr table2.field2
```

The **LEFT JOIN** and **RIGHT JOIN** have these parts:

Part	Description
table1, table2	The names of the tables from which records are combined.
field1, field2	The names of the fields that are joined. The fields must be of the same data type and contain the same kind of data, but they do not need to have the same name.
Comp_opr	Any relational comparison operator: "=," "<," ">," "<=," ">=," or "<>."

Self-join:

- ✓ A **self-join** occurs when a field in a table is related to another, or the same, field in the table.
- ✓ The classic example of a **self-join** is a table containing employee records. Very often this table will include a field specifying an individual's supervisor. But, because supervisors are also employees, the supervisor records are contained in the same table.

Syntax:

```
SELECT ...
FROM table1 { T1 | [AS T1]} , table1 { T2 | [AS T2]}
WHERE ...
```

Assignment (II-4)

Using the **Suppliers-Parts-Shipments** database (**Suppliers-Parts-Shipments.accdb**) which has been created in the previous laboratory, do the following queries:

1) Get all combinations of supplier and part information? (**Cartesian Product**)

2) Get full details of S & P that are located at the same city? (**equi-join**)

3) Make the **Natural Join** between S & P on city column?

4) Get all combinations of S & P such that supplier city follows the part city in alphabetical order?

5) Get SID and PID combination such that the supplier and part are **col-located** (i.e. at the same city), omitting supplier with status 20?

6) Get part ID and quantity supplied by Smith?

7) Same as **Q6**, but use (**using Inner Join**)?

8) Get the supplier ID/IDs for the supplier(s) that does/do not supply any part (i.e there is no shipment for him/them)? (**using Left Join**)

9) Get part name that supplied by Jones?

10) Get SID for suppliers who col-located with 'S2'? (**Self-Join**)

❖ Union Query:

- ✓ **Union Query** used to combine sets of records from different tables with common fields.
- ✓ So, creates a **Union Query**, which combines the results of two or more independent **queries**, **SELECT statements** or **tables**, as described in Figure (5-1).

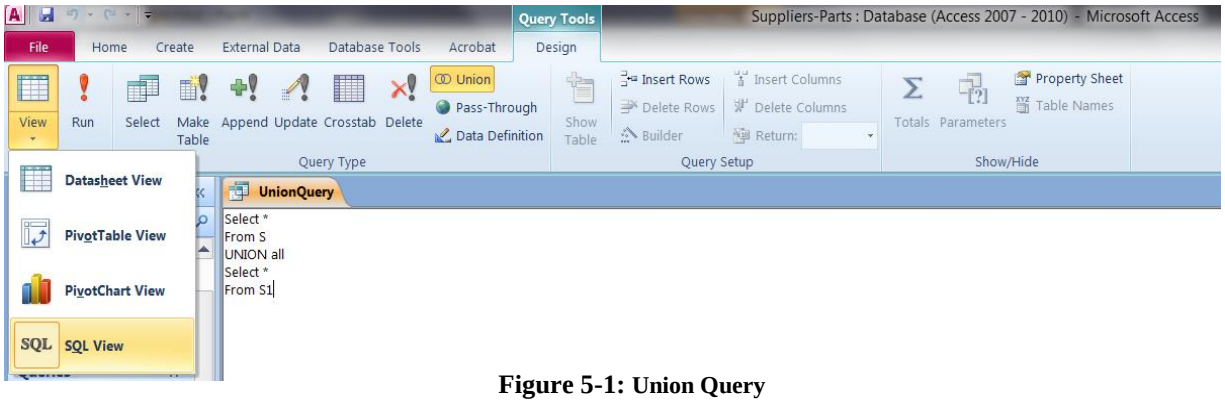


Figure 5-1: Union Query

UNION Statement:

Syntax:

[TABLE] ***query-1***

UNION [ALL]

[TABLE] ***query-2***

[UNION [ALL]

[TABLE] ***query-n*** **[…]]**

The **UNION** statement has these parts:

Part	Description
query1-n	A SELECT statement, the name of a stored query, or the name of a stored table preceded by the TABLE keyword.

Notes:

1- **Union Query** is an **SQL Specific Query** is one that can be created only by writing an SQL statement in **SQL view**.

2- You can merge the results of **two** or **more queries**, **tables**, and **SELECT statements**, in any combination, in a single **Union Query**.

3- By default, **no duplicate** records are returned when you use a **Union Query**; however, you can include the **ALL** predicate to ensure that all records are returned. This also makes the query run **faster**.

4- All queries in a **Union Query** must request **the same number of fields**; however, the fields do not have to be of **the same size** or **data type**.

❖ Select Query with Special Operators:

You use the special operators such as Is Null, Like, Between .. And, In, etc., with SELECT statement to return a result as described in the following table.

OPERATOR	PURPOSE	EXAMPLE
Is Null	**Determines whether a value is Null or Not Null.**	**Mobile Is NOT Null - OR - Sid Is Null**
Like 'pattern'	**Matches string values by using the wildcard character ? or ***	**Sname NOT Like 'J*' - OR - Sname Like '*a*'**
Between value1 And value2	**Determines whether a *numeric* or *date* value is found within a range.**	**Qty NOT Between 100 And 400 - OR - LoginDate Between #01-04-2015# And #01-05-2015#**
In (value1 , value2 ...)	**Determines whether a value is found within a set of values.**	**Color NOT In ('red', 'green', 'blue') - OR - Status In (15, 18 , 25)**

Tip: you can use **Not** operator before **Like**, **Between...And**, **In** as the syntaxes below:

expr [Not] Like 'pattern'

expr* [Not] Between *value1* And *value2

***expr* [Not] In (*value1, value2, …*)**

but with Is after that. **expr Is [Not] Null**

❖ Subquery (Or Inner Query):

✓ A **Subquery** is a **SELECT** statement nested inside a

- **Select**, **Make-Table**, **Insert**, **Delete**, or **Update** statement or inside another
- **Subquery**.

✓ A **Subquery** is usually added in the **Where** clause of the **SQL** statement.

✓ Most of the time, a **Subquery** is used when you know how to search for a value using a **SELECT** statement, but do not know the exact value in the database.

Notes:

1- A **Subquery** is also called an **inner query**, while the **SQL Statement** containing a **Subquery** is also called an **outer query**.

2- **SQL Statements** that include a **Subquery** usually take one of these formats:

```
SQL Statement
WHERE expression [NOT] IN (Subquery)
```

Or ;

```
SQL Statement
WHERE expression Comparison_Operator (Subquery)
```

Assignment (II-5)

Using the **Suppliers-Parts-Shipments** database (**Suppliers-Parts-Shipments.accdb**) which has been created in the previous laboratory, do the following queries:

1) Get PID for parts that either weight more than **16** pound or are supplied by '**S3**'? (**Using Union**)

2) Write the SQL code to get the following **Recordset:**

The Result
Smith
Jones
Blake
Clark
Adams
Nut
Blot
Screw
Screw
Cam
Cog

3) Get the full details of shipments that have quantities between **200** and **350**?

4) Retrieve the full details of all suppliers that are not in **London** city? (**Using in**)

5) Get the full details of suppliers that SIDs are '**S2**' , '**S4**' and '**S5**'?

6) Retrieve the supplier name for suppliers who supply '**P2**'? (**Using Subquery**)

7) Get the part name for the parts supplied by **Clark**? (**Using Subquery**)

8) List SID for suppliers with status value less than the current **maximum** status in **S** table?

9) Get SID for suppliers who col-located with '**S2**'? (**Using Subquery**) ***(Self-Join)***

10) Retrieve all available information for all parts supplied by supplier whose minimum status value in **S** table?

❖ Action Query:

- ✓ So far, the queries we've covered are typical select queries. As the name implies, a select query selects records from a single table/query, or a number of tables/queries, and arranges the selected records as a **recordset**.

- ✓ The **recordset** is then used by the application, most often as a **data source** for a **form** or **report**. But in most situations, there is often a need to perform an **action** on a group of records, such as **deleting** inactive supplier from the *suppliers* table, **updating** the quantity in every record in the *shipments* table, or **adding** a new part to *parts* table etc.

- ✓ So, an **Action Query** is a type of query that performs an **action** (delete, update, insert and so on) against a group of records.

- ✓ MS **Access** provides **four** types of **Action Query** (**Make-Table**, **Delete**, **Update**, **or Append**), as shown in Figure (6-1):

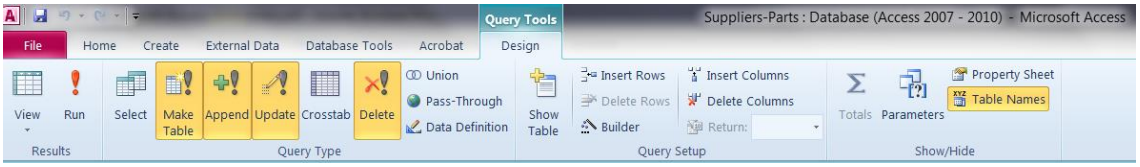


Figure 6-1: Action Queries

> **Tip:** in MS Access, **Action Query** can't be created using the **Query Wizard**.

1) SELECT… INTO Statement (Make-Table Query):

Makes a **new table** from the selected records. For example, you may want to create an archive table of all inactive supplier records.

> **Syntax:**
>
> **SELECT** ***field1*** **[**, ***field2*** **[**, **…]]**
>
> **INTO** ***newtable*** **[IN** ***externaldatabase*** **]**
>
> **FROM** ***source***

The **SELECT…INTO** statement has these parts:

Part	Description
field1, field2	The name of the fields to be copied into the new table.
newtable	The name of the table to be created. It must conform to standard naming conventions. If *newtable* is the same as the name of an existing table, a trappable error occurs.
externaldatabase	The path to an external database.
source	The name of the existing table from which records are selected. This can be single or multiple tables or a query.

Note: You can use **Make-Table Query** to archive records, make backup copies of your tables, or make copies to export to another database.

2) DELETE Statement (Delete Query):

- ✓ The **Delete Query** is used to delete the existing records from a table.
- ✓ You can use **Where** clause with **Delete Query** to delete selected rows, otherwise **all** the records would be deleted.
- ✓ Extending the previous example, when the inactive suppliers have been archived as separated table (i.e. you did **Make-Table Query**), you might use a **Delete Query** to remove the inactive supplier records from the *Suppliers* table.

Syntax:

```
DELETE [table.*]
FROM table
WHERE criteria
```

The **DELETE** statement has these parts:

Part	Description
table	The optional name of the table from which records are deleted.
table	The name of the table from which records are deleted.
criteria	An expression that determines which records to delete.

Notes:

1- You can use **Delete Query** to remove records from tables that are in a **one-to-many** relationship with other tables. ***Cascade delete*** operations cause the records in tables that are on the many side of the relationship to be deleted when the corresponding record in the one side of the relationship is deleted in the query.

2- A **Delete Query** deletes **entire** records, not just data in **specific fields**. If you want to delete values in a specific field, create an **Update Query** that changes the values to **Null**.

3) UPDATE Statement (Update Query):

✓ The **Update Query** is used to modify the existing records in a table. For example adjusting part's weight or supplier's status.

✓ You can use **Where** clause with **Update Query** to update selected rows otherwise **all** the rows would be affected.

Syntax:

UPDATE ***table***
SET ***newvalue***
WHERE ***criteria***

The **UPDATE** statement has these parts:

Part	Description
table	The name of the table containing the data you want to modify.
newvalue	An expression that determines the value to be inserted into a particular field in the updated records.
criteria	An expression that determines which records will be updated. Only records that satisfy the expression are updated.

Notes:

1- You can change several fields at the same time, using comma (,).

2- **Update Query** is especially useful when you want to change many records or when the records that you want to change are in multiple tables (using ***Cascade update***).

4) INSERT INTO Statement (Append Query):

✓ Adds a record or multiple records to a table. This is referred to as an **Append Query**. For example, add a new part to Parts table.

✓ There are **two** basic syntaxes of **Append Query** as follows:

Syntax(1):

Single-record:

INSERT INTO ***target*** **[(*field1* [, *field2* [, …]])]**
VALUES (*value1* [, *value2* [, …]])

Notes:

1- In first case, **Syntax (1)**, if your SQL code specifies the name and value for each field of the target table. You must specify each of the fields of the table that a value is to be assigned to and a value for that field. When you do not specify each field, the **default value** or **Null** is inserted for **missing columns**.

2- You may not need to specify the field(s) name (**omit the field list**) in the **Append Query** if you are adding values for **all** the fields of the table; But make sure the order of the values is in the same order as the fields in the table (for logical insert and datatype) ; otherwise, the **INSERT** operation will fail.

3- Be careful when you try to add a new value to field(s) in table has **constraint** (**primary key, foreign key, unique etc.**) in that field(s). For example, make sure you append **unique**, **non-Null** values to the primary key field(s); if you do not, the Microsoft Access database engine will not append the records.

Syntax(2):

Multiple-record:

INSERT INTO *target* **[(***field1* **[, *field2* [, ...]])] [IN** *externaldatabase***]**

SELECT [*source***.]***field1* **[, *field2* [, ...]]**

FROM *tableexpression*

The **INSERT INTO** statements have these parts (**Both syntaxes 1 & 2**):

Part	Description
target	The name of the table
field1, *field2*	Names of the fields to append data to, if following a *target* argument, or the names of fields to obtain data from, if following a *source* argument.
externaldatabase	The path to an external database.
source	The name of the table or query to copy records from.
tableexpression	The name of the table or tables from which records are inserted. This argument can be a single table name or a compound resulting from an INNER JOIN, LEFT JOIN, or RIGHT JOIN operation or a saved query.
value1, *value2*	The values to insert into the specific fields of the new record. Each value is inserted into the field that corresponds to the value's position in the list: *value1* is inserted into *field1* of the new record, *value2* into *field2*, and so on. You must separate values with a comma, and enclose text fields in quotation marks (' ').

Note:

You can also use **Append Query** to append a set of records from another table or query by using the **SELECT … FROM** clause as shown above in **Syntax (2)**. In this case, the **SELECT … FROM** clause specifies the fields to append to the specified target table.

Assignment (II-6)

First of all, make a backup (copy) of your database (**Suppliers-Parts-Shipments-backup.accdb**), and then do the following queries:

1) Compute the total quantities for each part and stored it with its associate name in a **Temp1** table.

2) Store in **Temp2** table, all part names which are supplied by the supplier, its name will be given during run-time, use [Please, enter the supplier's name] as name of parameter.

3) Modify the status of **Blake** to null value.

4) Add to database the information for the new supplier (Ahmed) who located at (Baghdad), give him a supplier ID (S6) and a status (50).

5) Insert the following data to **S** table {S4, Ali, 25, Basra}. ***(Hint: you will encounter an error!) Why?***

6) The supplier has ID = S6 supplied 150 pieces of p2, add this information to database.

7) Append the following data to **P** table at run-time:

[Please, enter the part's ID :] ⇒ P10
[Please, enter the part's name :] ⇒ Roller
[Please, enter the part's color :] ⇒ Gray
[Please, enter the part's weight:] ⇒ 35
[Please, enter the part's city :] ⇒ London

8) Add the following data to **SP** table (S5, P9, 600). ***(Hint: you will encounter an error!)! Why?***

9) Double the statues of all suppliers in '**Paris**'.

10) Set the shipment quantities to ***Zero*** for all parts with 'red' color.

11) **Smith** no longer supply parts from '**London**', erase all these shipments.

12) **Jones** supplied another **25** pieces of '**Nut**', add this information to our database.