

ACCOUNTING

PLUS ONE

(FOSS Tools for Accounting)

PART III - DRAFT Version 1.0



Prepared by :

Kerala Infrastructure & Technology For Education

Poojappura, Thiruvananthapuram-12,

Website : www.kite.kerala.gov.in, email : contact@kite.kerala.gov.in

Phone : 0471-2529800, Fax: 0471-2529810

© Department of Education, Government of Kerala

For any queries, please contact the above Office.

PREFACE

Kerala has become the World's first State with the largest deployment of Free and Open Source Software in Education Sector, as part of the ICT enabled education being implemented by IT@School in High School section in the State. In August 2008, Government had instructed all institutions under General Education Department to strictly use Free Software alone in all future teaching and training activities and a Government Order in this regard is in effect. Similarly, the Information Technology Department has also released a Circular in July 2016, directing all departments to use FOSS. Adhering to these directions, the Computer Science and Humanities sections in the Higher Secondary had shifted to Free Software. But at the same time, in the Commerce section in Higher Secondary (Computerised Accounting System), there were using MS Excel, MS Access and Tally package which are proprietary in nature.

The 48th meeting of the State Curriculum Steering Committee held on 15.02.2017, has decided to adopt Free & Open Source Software (FOSS) for entire activities in Higher Secondary Sections. The shift to Free Software in Higher Secondary is planned to be implemented without making any changes in the critical structure of the present syllabus, but by incorporating changes in the software which is being used. Accordingly, the revised textbook would include Free Software applications such as LibreOffice Calc, LibreOffice Base and GNUKhata which would replace the proprietary applications. IT@School would implement mechanisms for easy classroom transactions of chapters including customisation of applications, teacher training, video tutorials etc. This handbook is an attempt to help the higher secondary teachers and students in easy classroom transactions.

K.ANVAR SADATH

Vice Chairman & Executive Director
Kerala Infrastructure & Technology For Education

TERM 3

**ACCOUNTING SYSTEM USING
DATABASE MANAGEMENT SYSTEM**

LEARNING OBJECTIVES

After studying this chapter, you will be able to :

- *identify the resources of LibreOffice Base as DBMS;*
- *create data tables described in a data- base design and set relationship among these tables;*
- *explain the LibreOffice Base basics and procedures to create forms using LibreOffice Base;*
- *describe and create voucher forms in consonance with different database designs;*
- *identify information requirement of reports for querying databases;*
- *formulate and implement queries for retrieving data and information for presentation in accounting reports ;*
- *implement the process in LibreOffice Base for generating accounting reports by using accounting information queries.*

In chapter 14, you have learnt about the fundamentals of creating a database design in the context of accounting system. This chapter deals with the basics of LibreOffice Base for implementing the databases and specifically deals with implementation of accounting databases, the design of which has been shown, described and discussed in chapter 14 as Model-I and Model-II. The accounting database design has been discussed below in terms of its implementation modalities in the context of LibreOffice Base.

15.1 LibreOffice Base and its Components

It is one of the popularly used Database Management System (DBMS) to create, store and manage database. Every component that is created using LibreOffice Base is an object and several such similar objects constitute a class. LibreOffice Base is functionally available with the following four-object classes. Each of these object classes is capable of creating their respective object replicas.

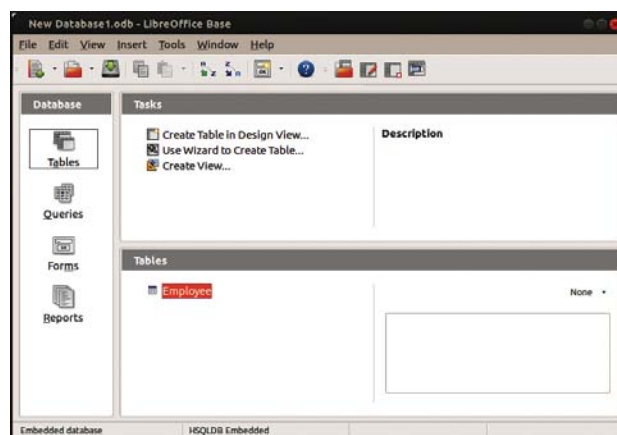


Figure 15.01 : An example of database window to work in LibreOffice Base Window

- ◆ **Tables** : This object class allows a database designer to create the data tables with their respective field names, data types and properties.
- ◆ **Queries** : This object class is meant to create the SQL compatible query statement with or without the help of Graphic User Interface (GUI) to define tables, store data and retrieve both data and information.
- ◆ **Forms** : This object class allows the designer to create an appropriate user interface to formally interact with the back end database, defined by the tables and queries.
- ◆ **Reports**: This object class is used to create various reports, the source of information content of which is based on tables, queries or both. Such reports are designed in LibreOffice Base according to the requirement of end-user.

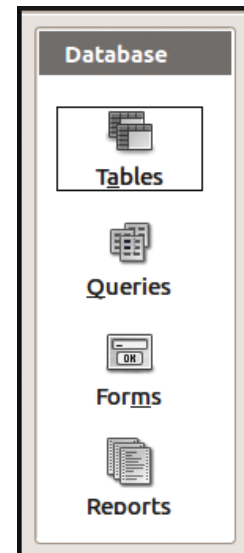


Figure 15.2
Object Classes

Each of these object classes is contained in the named database file of LibreOffice Base with **odb** extension. Whenever this file is opened, it opens with all the above object classes available on the left hand side. As and when the specific objects are created or designed, they get listed on right hand side of this window against each of these object classes.

Capabilities of LibreOffice Base

LibreOffice Base has certain capabilities, which bring it closer to an ideal Database Management System. These capabilities are :

- ◆ Storing the data in an organised manner.
- ◆ Enforcing data integrity constraints.
- ◆ Representing complex relationship among data.
- ◆ Providing for persistent storage of database objects.
- ◆ Restricting unauthorised access to database.
- ◆ Allowing fast retrieval of data with or without processing by using SQL.
- ◆ Flexibility to create multiple user interfaces.
- ◆ Providing for data sharing and multi-user transaction processing.
- ◆ Supporting multiple views of data and information.

15.1.1 Basics for Creating a Database in LibreOffice Base

When a new database is created from the scratch, there is complete control over the database objects, their properties and the relationships. In order to create a new database, the following steps are required :

- ◆ Open LibreOffice Base
- ◆ You will be taken to the 'Database Wizard' Screen as shown in Figure-15.3. **Select Database**

helps you to create a database file or open an existing database file or connects to an existing database. Depending on the type of operation and the type of database, the Database Wizard consists of a varying number of steps.

- ◆ “Create new database” option leads to registration of the new database.

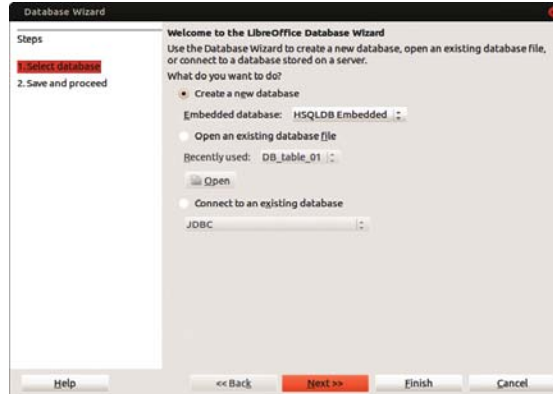


Figure 15.3 : Database Wizard

LibreOffice Base responds by displaying Save New Database dialog box, which prompts the designer to enter a file name and a location for the database.

15.1.2 Creating Tables in LibreOffice Base

The creation of tables in LibreOffice Base requires the following steps.

- ◆ Click at **Tables** object of LibreOffice Base left pane, followed by clicking at **Create Table in design view**. This results in providing a table window, the upper part of which has three columns: **Field Name**, **Field Type** and **Description**. It is meant to define the schema of a table being created. Each of its rows corresponds to a column of the table being created. Two primary properties of the column of a table are its field name and field type.

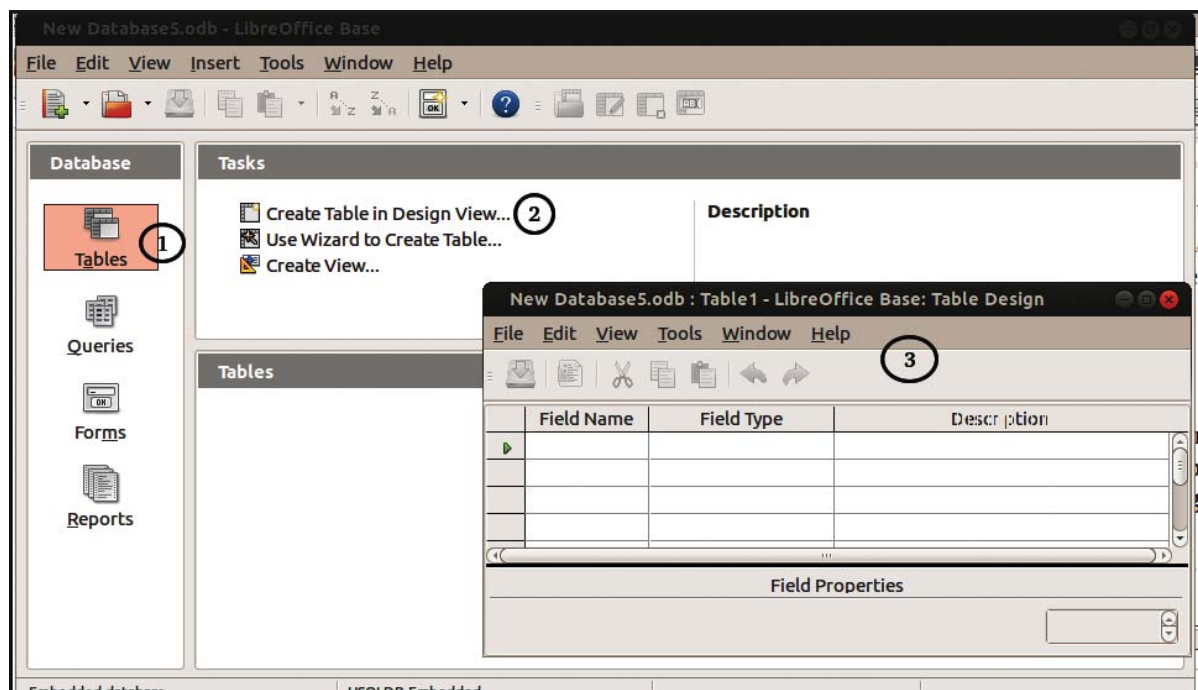


Figure 15.4 : Table Creation

- (a) **Field name** : refers to column name of the table being created. The name of the column should be a string of contiguous characters. The Field name is meant to define the name of column to be created, followed by field type of such column. The designer can optionally provide description of the column also. Once the field type is defined, the designer can further specify the properties of each column in the lower part of the Table window.
- (b) **Field Types** : LibreOffice Base supports different data types, Some useful data types are :

Text [VARCHAR]: It is used to store a string of characters, up to the specified number of characters. words or numbers that are not to be used in any arithmetic calculations. It is the default data type because of being used most frequently. There are different text data types such as VARCHAR, VARCHAR_IGNORECASE, and CHAR.

Memo [LONGVARCHAR]: A Memo field can be used for storing large amounts alphanumeric information. Some typical uses for this data type would be a note, comments, description, or address field. But a field with this Field Type is not amenable to sorting or filtering of data records.

Field Name	Field Type
Emp_id	Text [VARCHAR]
	Text (fix) [CHAR]
	Number [NUMERIC]
	Decimal [DECIMAL]
	Integer [INTEGER]
	Small Integer [SMALLINT]
	Float [FLOAT]
	Real [REAL]
	Double [DOUBLE]
	Text [VARCHAR]
	Text [VARCHAR_IGNORECASE]
	Yes/No [BOOLEAN]
	Date [DATE]
	Time [TIME]
	Date/Time [TIMESTAMP]
	OTHER [OTHER]

Figure 15.5 : Field Type

Number : It is meant to store numbers, which could be integers single (to store values with decimal point up to a certain limit), double (to store values in decimal point with greater magnitude and more precision) or decimal types.

Date : Used to store date into Database table.

Time : It is used to store time only.

Date/Time : It is used to store combination of both.

Currency : It is used for storing numbers in terms of Dollars, Rupees or other Currencies.

Yes/No : It is to declare a logical field which may have only one of the two opposite values alternatively given as: Yes or No, On or Off, True or False.

Field Name	Field Type	Description
empid	Number [NUMERIC]	Employees ID
empname	Text [VARCHAR]	Employees Name
designationid	Text [VARCHAR]	Designation of Employee
add	Text [VARCHAR]	Address of Employee
dob	Date [DATE]	Date of Birth

Field Properties of selected field	
Entry required	No
Length	100
Default value	
Format example	@

Figure 15.6 : Data Type and Properties

Other Data types : Different Numeric Types like double, integer, etc and Alphanumeric Types, images and serialized Java objects can also be stored in LibreOffice Base.

(c) **Properties :** In a database, field properties play an important role in controlling the behaviour of a field. In BASE, you can assign the following types of field properties:

- **Entry Required:** Specifies whether the entry of data in the selected field is mandatory or not. For example, in case you set the value as Yes in the field property of a selected field, it means that you must enter data in the field. On the other hand, if the value is set as No, then you need not enter any data in the field. In other words, you can skip this field.
- **Length:** Specifies the field length, that is, the size of the field.
- **Decimal Places:** Specifies the place of the decimal point from the right side of a numeric value. For example, if you specify the value 3 in the Decimal Places field property, it means that the decimal point is placed at the fourth position from the right side of the numeric value.
- **Default Value:** Specifies the value that is added automatically in a field. You can change this value with a value of your own.
- **Format Example:** Allows you to control the appearance of the data in a table. You can use the built-in formats available in BASE or define your own formats.

Primary Key : After defining all the columns of the table, the primary key column of the table can be specified as any of the columns that are expected to have unique data values. Perform the following steps to set the primary key in the Student table:

- ◆ Right clicking at the field to be specified as primary key
- ◆ Select primary key item from the context menu (Figure 15.7).
- ◆ The primary key is set for the selected field in the table.

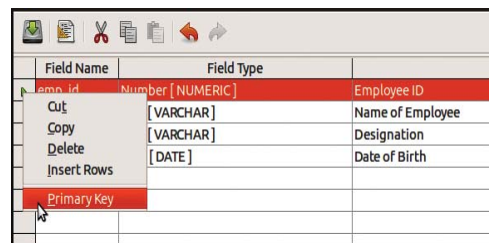


Figure 15.7 : Setting Primary Key

Save the table design by clicking at File item of menu bar followed by click at Save option. LibreOffice Base responds by providing a generic default name of table “Table1”. The table name provided by LibreOffice Base may be accepted by clicking at OK or changed by re-typing another name at the input dialog box. This must be followed by clicking OK. The table stands created and appears as listed to the right of table object. Every other table, which constitutes part of the database design, may also be created in the same manner as described above.

The foregoing discussion in this chapter is divided into four sections: Creating tables and relationships for accounting databases; Vouchers and forms; information using queries and generating accounting reports.

15.2 Creating Tables and Relationships for Accounting Database

The implementation of each database design is conditioned by its particular table structure and its interrelationships. Such implementation modalities have been discussed in detail for various types of transaction vouchers.

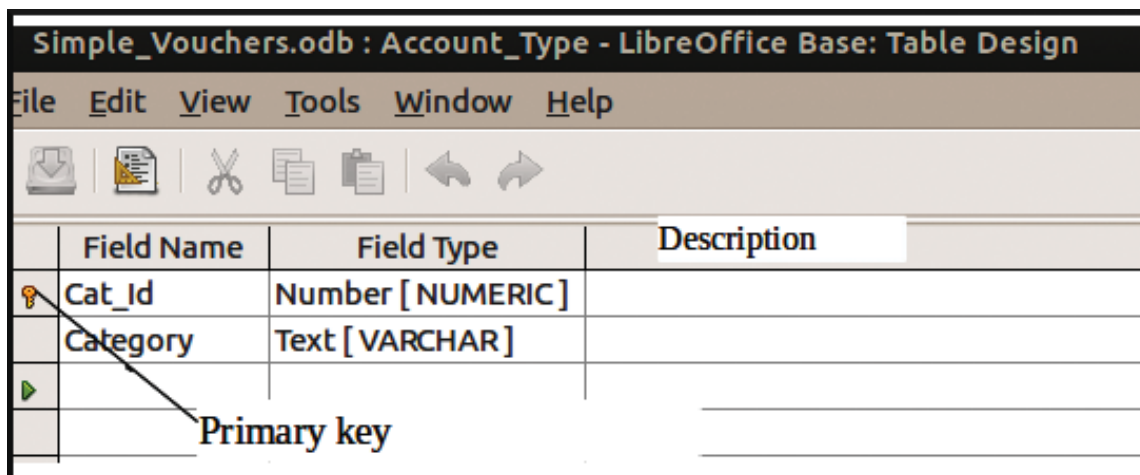
15.2.1 Database Design for Simple Transaction Vouchers

Suppose you are designing an accounting database by using simple transaction vouchers. Here we are considering five data tables: Account_Type, Accounts, Employees, Vouchers and Support. For the purpose of implementation, each table is described below in terms of their storage structure, i.e. Field names, Field types and Field properties:-

(a) Account_Type : This table has two columns: Cat_Id and Category.

- **Cat_Id** : This column of the Account Type table is meant to specify the identification value of the category of accounts. Since there are limited number of accounts type and are being expressed as numeric only, the field type of this field can be taken as Number (NUMERIC).
- **Category** : This field is meant to store the string of characters to express the category of account such as Expenses, Revenues, Assets and Liabilities. Its field type should be Text (VARCHAR) with suggested field size set to 15 characters.

The table creation screen is shown below (Figure 15.6).



	Field Name	Field Type	Description
Key	Cat_Id	Number [NUMERIC]	
	Category	Text [VARCHAR]	

Primary key

Figure 15.8 : Database Table Design for the table Account_Type

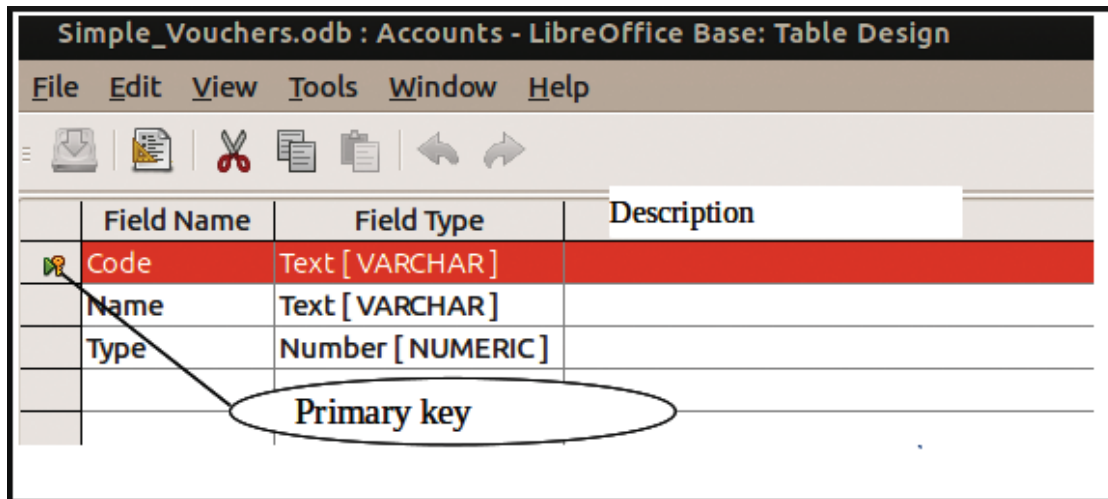
b) Accounts : This table has three columns: Code, Name and Type.

- **Code** : A unique account number or code identifies an account. This column is meant to store this code. Its field type is chosen as Text (VARCHAR), because it is not to be subjected to any calculations, with suggested field size set to 6 characters. Because of uniqueness in values, this field is a good primary key field. Set field properties, 'Entry Required' as 'Yes'.
- **Name** : In a system of accounting, every account has a name. This column is meant to store the name of an account corresponding to the account code by which it is identified.

Its field type is declared as Text (VARCHAR) because it is a string of characters not required for any calculations. Its field size may be set to 30 characters, which is considered to be long enough to accommodate the name of account.

- **Type** : Every account must belong to one of the accounts type as stored in Account Type table. This field is a foreign key to reference Cat_Id field of Account Type table. Its data type and other properties must be the same as that of Cat_Id field in Account Type table.

The table creation screen is shown below (Figure 15.9)



Field Name	Field Type	Description
Code	Text [VARCHAR]	
Name	Text [VARCHAR]	
Type	Number [NUMERIC]	

Figure 15.9 : Table Design for the Table Accounts

(c) **Employees** : This table stores the data pertaining to employees of the organisation and is designed to have following six columns :

- **Emp_Id** : Each employee is identified by a unique data value called EmpId, which in turn gets reflected in employee table as a column to store for each employee record a unique identification value. The field type of this column is Text (VARCHAR) with field size equal to 4. Being a column to store unique values and also because of its capability to identify an employee record, it is designated as primary key field. Set field properties, 'Entry Required' as 'Yes'.
- **F_name** : This column refers to the first name of employee and its data type is declared as Text (VARCHAR) because it is meant to store string of alphabets. Its Field size is set to 10 on the assumption that first name of every employee can be completely accommodated within this field size.
- **M_name** : Mname column is meant to store the middle name of an employee. Its field type is declared as Text (VARCHAR) with field width equal to 10. The Entry Required Property should be 'No' to imply that many employees may not have middle name.
- **L_name** : Lname column has been included in the table structure to store the Last name of an employee. The data type of this column is Text (VARCHAR) with field size set to 10.
- **Phone_no** : This column is meant to store the Phone number of the employee and its field type is set to Text (VARCHAR) with field size equal to 12.
- **Super_Id** : This column in the Employee table structure refers to EmpId of the supervisor or immediate superior of the employee. Its field type is set to Text (VARCHAR) with field width 4, the same as is for Emp_Id.

The table creation screen appears as follows (Figure 15.10) :

(d) **Vouchers** : This table has been designed to store the transaction data as contained in a voucher.

It has eight columns, the details of each are given below :

Field Name	Field Type	Description
Emp_Id	Text [VARCHAR]	
F_name	Text [VARCHAR]	
M_name	Text [VARCHAR]	
L_name	Text [VARCHAR]	
Pphone_no	Text [VARCHAR]	
Super_Id	Text [VARCHAR]	

Figure 15.10 : Database Table Design for the table Employee

- V_no** : This column is meant to store voucher number, which indicates the distinct identity of a transaction. Its field type may be Number (NUMERIC), if numeric digits are only assigned to each of the vouchers. However, its field type is normally taken as Text (VARCHAR) because it is amenable to any type of numbering, coding or ordering scheme: numeric, alpha-numeric or formatted reference. Its width may be set to 6 so that first 2 places to the left refer to numeric month of the date and next 4 places to numeric digits giving identity to each of the transactions that have occurred during the month under reference. This column is designed to have distinct values and therefore can be designated as primary key of the table. Accordingly, its value cannot be null and therefore set field properties, 'Entry Required' as 'Yes'.
- Debit** : This column is meant to store the code corresponding to an account, which has been debited in recording a transaction. Since it references the code column, which is the primary key of Accounts table as described above, it is a foreign key column in Vouchers table. The field type and properties of this column should be the same as that of code column of Accounts table.
- Amount** : This column is meant to store the amount of transaction and is common to the accounts being debited and credited. Its field type can be Number (NUMERIC) decimal places may be set to 2.
- V_date** : This column of the table stores the date of transaction. Its field type is set to Date (DATE).
- Credit** : This column is meant to store the code corresponding to the account being credited in recording a transaction. Like Debit column, this column too shares the same properties as code column of Accounts table and must also be dealt with in the same manner as Debit column described above.
- Narration** : This column is meant to store the narration. Its field type can be set to Text (VARCHAR) with field size set to 100 characters.
- Prep_by** : This column is meant to store the identity of an employee who has prepared the voucher. Emp_Id as defined and described in schema of Employees table identifies the employee. The data type of this field and other properties must be identical to that of Emp_Id.

- **Auth_by** : This column is meant to store the identity of the employee who has authorised the vouchers. This column is similar to Prep_by column. Therefore, its field type and properties with Emp_Id are the same as those for Prep_by column.

The table creation screen appears as follows (Figure 15.11) :

Field Name	Field Type	Description
V_no	Text [VARCHAR]	
Debit	Text [VARCHAR]	
Amount	Number [NUMERIC]	
V_date	Date [DATE]	
Credit	Text [VARCHAR]	
Narration	Text [VARCHAR]	
Prep_by	Text [VARCHAR]	
Auth_by	Text [VARCHAR]	

Figure 15.11 : Database Table Design for the table Vouchers

(e) Support : This table is created to store the details of support documents annexed to a voucher.

It is designed to have the following four columns:

- **V_no** : This column is meant to store the voucher number to which this document is annexed. Its field type should be the same as that of V_no in Vouchers table. Its value cannot be null and therefore set field properties, 'Entry Required' as 'Yes'. Since there may be more than one support documents annexed to a voucher, the values stored in this column cannot be unique and therefore this column alone cannot be primary key field.
- **S_no** : This column has been included in the table structure to store serial numbers 1,2,3 to ... correspond to the serial number of documents being annexed. Duplicate values will occur in this field also because the serial number of documents across the vouchers shall be the same. However, both the columns: V_no and S_no together provide a unique value because the documents, for every voucher are serially numbered and therefore unique. Both the columns together need be declared as Primary key of this table.
- **D_name** : This column refers to Document name. Its field type is Text(VARCHAR) with field size equal to 30 to mean that within this character limit the document name can be suitable accommodated.
- **S_date** : This column refers to any date reference given in the support document. Its field type is Date(DATE).

The table creation screen appears as follows (Figure 15.12) :

Field Name	Field Type
V_no	Text [VARCHAR]
S_no	Text [VARCHAR]
D_name	Text [VARCHAR]
S_date	Date [DATE]

Figure 15.12 : Database Table Design for the table Support

15.2.2 Modified Design for Implementing Compound Vouchers (Creating Relationships)

Suppose, there are two voucher tables: Vouchers_Main and Vouchers_Details, used for the data base, with the following fields:

(a) **Vouchers_Main** : This table has been created to store one record for every transaction. It consists of 6 columns such as V_no, Acc_Code, V_date, Prep_by, Auth_by and Type.

- **Acc_Code** : This column is meant to store the complementing account code, which in the context of debit voucher is credit account and in the context of credit voucher is a debit account. Its field type and properties must be the same as that of Code column of Accounts table.
- **Type** : This column has been created to store a value 0 (for debit voucher) or 1 (for credit voucher). Its type therefore is set to Number (NUMBER). This column is very important and therefore its values must be carefully stored and interpreted in preparing accounting reports. The field types and properties of V_no, V_date, Auth_by and Prep_by continues to be same as have been defined and discussed in Vouchers table of Simple Vouchers Design.

The table creation screen is shown below (Figure 15.13) :

Field Name	Field Type	Description
V_no	Text [VARCHAR]	
Acc_code	Text [VARCHAR]	
V_date	Date [DATE]	
Prep_by	Text [VARCHAR]	
Auth_by	Text [VARCHAR]	
Type	Number [NUMERIC]	

Figure 15.13 : Database Table Design for the table Voucher_Main

(b) **Vouchers_Detail** : This table is meant to store those data items of the voucher, which appear in the grid of debit or credit vouchers. However, the Total amount of voucher is not stored because it is derived data. It consists of V_no, S_no, Code, Amount and Narration as its columns.

- **V_no** : This column is meant to store voucher number of Debit/Credit record of Vouchers_Main

table to which the Credit/Debit entries of vouchers_Details table are related. Its field type should be the same as that of V_no in Vouchers_Main table.

- **S_no** : This column has been included in the table structure to store serial numbers 1,2,3... to correspond to the serial number of debit/credit entries.
- **Code** : This column is meant to store the account codes, which in the context of debit voucher are debit accounts and in the context of credit voucher are credit accounts. Its field type and properties must be the same as that of Code column of Accounts table.

The data type and properties of Amount and Narration column continue to be the same as already described and discussed for Vouchers table.

The table creation screen appears as follows (Figure 15.14) :

	Field Name	Field Type	Description
	V_no	Text [VARCHAR]	
	S_no	Text [VARCHAR]	
	Code	Text [VARCHAR]	
	Amount	Number [NUMERIC]	
	Narration	Text [VARCHAR]	

Figure 15.14 : Database Table Design for the table Voucher_Details

The Voucher-main and Voucher_details are two distinct tables. Is there any relation between these tables? No.

Then how can we prepare a voucher report displaying details from both the tables? For this we should link these tables. For linking tables, we should identify a common field in both tables. Here, we can use V_no as the common field(linking field). Remember that the field type of linking fields should be same. The procedure of linking table is illustrated below

1. Click on Tools
2. Click on Relationships

Then, a window (Add tables) showing all tables appear. Click on the Voucher_main and click on add button. Also click on Voucher_details and click on add button. Click on close button on the window.

3. Click on New relation icon (icon below the Tools tab). Then, a new window named 'Relation' appear.

The relation design screen appears as follows (Figure 15.15) :

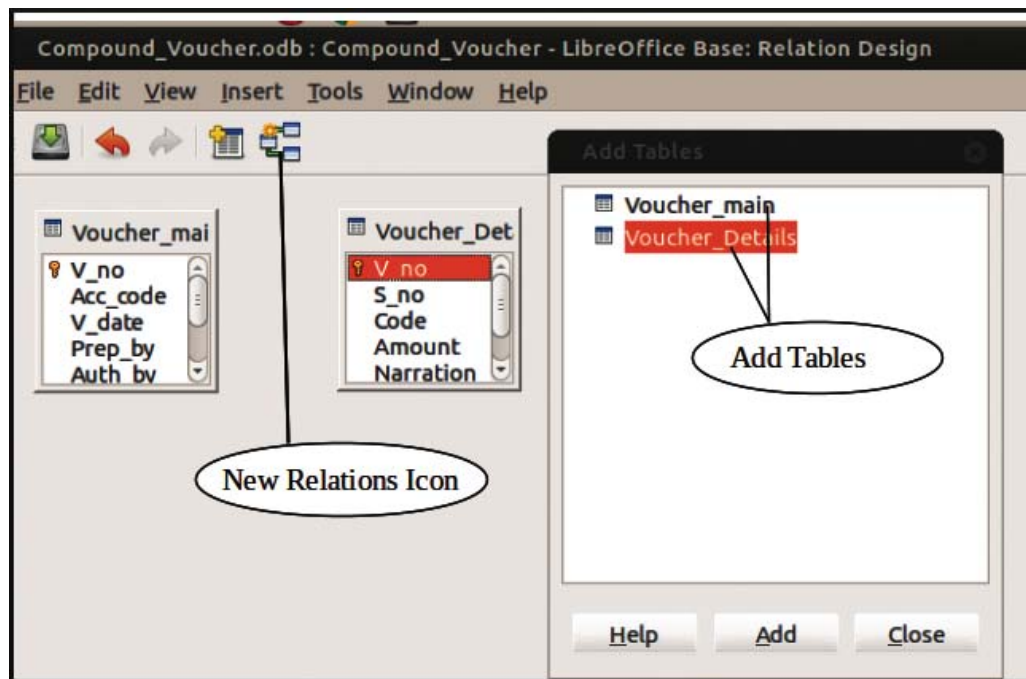


Figure 15.15 : Relation Design Window

4. Select V_no from both Voucher_main and Voucher_details (from the head fields involved). Then, click on update cascade and delete cascade. Also click on OK. A line linking the tables appear showing relationship.

Following figures shows the given steps in creating relations between the two tables (Figure 15.16):

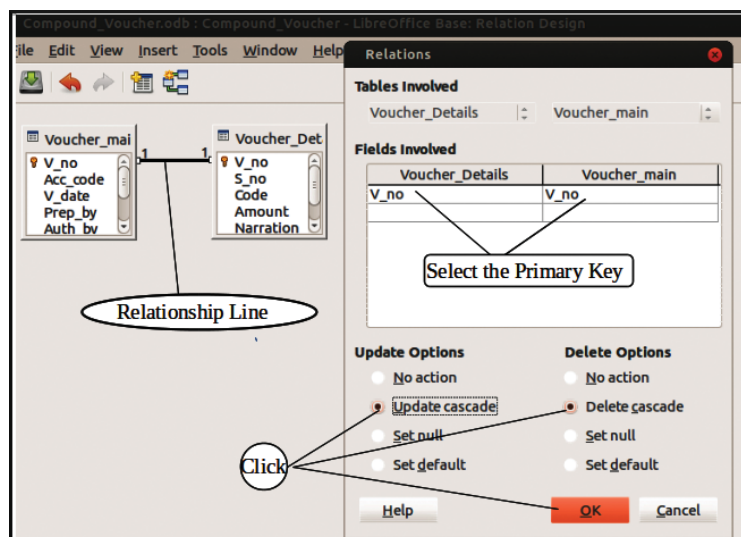


Figure 15.16 : Setting relation between two tables

15.3 Vouchers Using Forms

The scope of this section includes the basics for creating forms in LibreOffice base; transforming the voucher designs; and also the procedure for creating forms for vouchers.

15.3.1 Basics for Creating Forms in LibreOffice base

In LibreOffice base, it is possible to use forms for the creation, display and modification of records.

Data entry in a form is visually easier to understand than in a table, and is simpler for the user. A form may be designed, developed and used for the following purposes :

- ◆ Data Entry: Form is used for entering, editing and displaying data.
- ◆ Application flow : Form is used for navigating through an application.
- ◆ Custom Dialog Box : It can be used for providing messages to the user.

This is contrary to the belief that forms can be used only for data entry. The most common use of a form is to display and edit existing data and also for adding new data records.

15.3.2 Toolbar and form Controls

A toolbar is a collection of visual objects (or controls) that are placed (or embedded) on the form to provide some meaning or functionality. The form is designed by placing several such controls, which have their own functionality and properties.

15.3.3 Properties of Form Controls

Once a form has been created, it can be filled with visible controls. Some controls allow the content of the database to be displayed, or data to be entered into the database. Other controls are used exclusively for navigation, for searching, and for carrying out commands (interaction). Every form control is a complete object with its independent set of properties, which determine the shape, size, behaviour and functionality of the object. The properties of these objects are divided into three categories: **General, Data and Events**. All these properties may not apply to all the controls. Some important properties of these objects are as described below :

a) General Tab: This tab enables to define the general properties of a form control. Some of the important properties are as described as under (Table 15.1):

Control Properties	Use
Name	Allows to provide a name to control
Label field	Specifies the source for the label of the control.
Max.Text Length	The maximum number of characters that the user can enter
Visible	Specifies whether the control will be visible in live mode
Enabled	If (Yes), the user will be able to use the control field otherwise it will be displayed in a gray color.
Read-only	Set Yes will exclude any modification of the value
Printable	Specifies whether the control field to appear in a document's printout.
Tab stop	It determines a control field can be selected with the tab key or not.
Tab Order	It specifies the activation sequence within the form.
Width	Width of the field.
Height	Height of the field.
Font	Font, font size, and font effects can be set here.

Tab stop	It determines a control field can be selected with the tab key or not.
Tab Order	It specifies the activation sequence within the form.
Width	Width of the field.
Height	Height of the field.
Font	Font, font size, and font effects can be set here.
Alignment	The alignment option are Left, Right, Centre.
Vert. Alignment	The alignment option are Top, Middle, Bottom.
Background colour	Background colour of the text field
Border	Framing option are without frame, 3D-Look and Flat.
Border colour	If there is a frame (Flat), its colour can be set here.
Default Value	Sets the default value for the control field
Default Text	Sets the default text for a text box or a combo box.
Decimal accuracy	It specifies the number of decimal places with numeric and currency fields
Scrollbar	Adds the scrollbar type that you specify to a text box.
Edit mask	By specifying the character code in pattern fields, you can determine what the user can enter in the pattern field.
Help Text	A descriptive text for the control field.

Table 15.1 : Control Properties

Data tab

The Data Tab allows to assign a data source to the selected control. The important options available in the Data Tab are Data field: Empty string is NULL: Entry required: Filter proposal:

Events tab

On the **Events** tab we can link macros to events that occur in a form's control fields. Depending on the control, different events are available. Only the available events for the selected control and context are listed on the **Events** tab page. The events used in a form's control are shown in the Figure 15.17.

15.3.4 Common Controls in LibreOffice Base.

LibreOffice Base provides for a number of controls and Form controls can be activated by clicking Toolbars from View menu in design mode. Some of the common controls important for designing a Form are discussed below : see figure 15.17.

Check box: Check boxes allow you to activate or deactivate a function in a form.

Text Box: This control is included in a Form to provide a blank area for entering the data with or without default values . In a form, text boxes display data or allow for new data input. For example, Blank space next to Amount label,, is a text box control to receive the value of amount of voucher

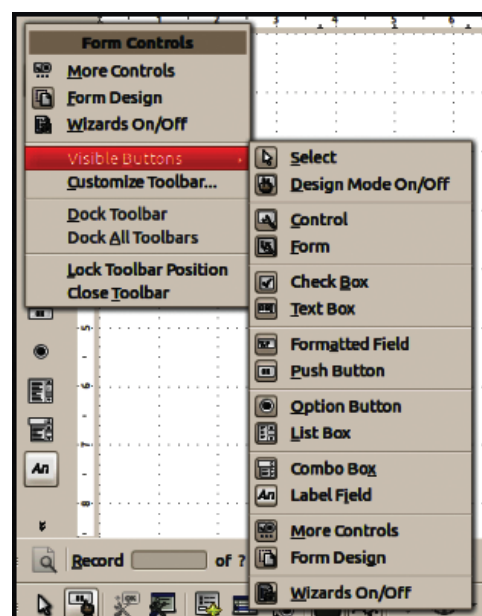


Figure 15.17 : Form Controls

Formatted field: A formatted field is a text box in which you can define how the inputs and outputs are formatted, and which limiting values apply. For eg. **Min. value** and **Max. value**: You can enter the minimum and maximum numeric value for a formatted field

Push Button: This function can be used to execute a command for a defined event, such as a mouse click. You can apply text and graphics to these buttons.

Option Button: Option buttons enable the user to choose one of several options. Option buttons with the same functionality are given the same name (Name property). Normally, they are given a group box.

List Box: In the case of a list box, the user selects one entry from a list of entries. These entries are saved in a database table and cannot be modified through the list box. The List Box Wizard will automatically appear after the list box is inserted in the document. This wizard helps you create the list box.

Combo box: In the case of combo boxes, users can select one entry from the list entries or enter text themselves. Combo boxes can also display the data of any table. In the case of a combo box, you can add additional text that can be written to the current database table of the form (values table) and stored there as desired. This control combines the features of a list box and text box by allowing a user to select an item from a list or enter a value using the keyboard.

Label Field: Creates a field for displaying text. These labels are only for displaying predefined text. Entries cannot be made in these fields. For eg Transaction Voucher, Voucher No, S.No, Debit, Credit, Amount, Narration, Authorised By, Prepared By etc.

15.3.5 Creation of Form

The Form can also be created via Use Wizard to Create Form or Create Form in Design View.

(a) Create Form by using wizard : The following procedure is followed for using the wizard to create a data entry Form :

Step 1 - From the main window ,click on **Forms** on the **Database pane**, and then select the **Use Wizard to Create Form** option. Immediately there is a window titled, **Form Wizard** which allows the designer to choose the data table along with the related available fields to choose from.

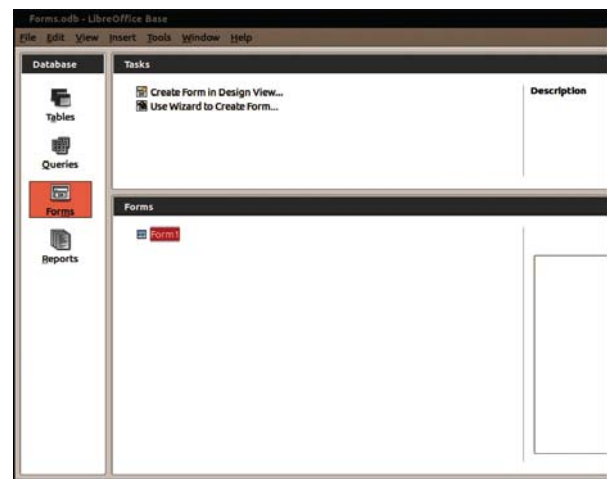


Figure 15.18: Form Selection from main window

Step 2: Select a table or a query that you want to work with. After the selection of the data source (table or query), the columns(fields) available are displayed on the List Box labelled **Available Fields**

Step 3: Field Selection - Select the desired fields by highlighting the field and clicking on the button with the single arrow pointing to the right. If all the fields are desired, click on the button with

two arrows pointing to the right. To remove selected fields from the second list box (Fields in the form) select the desired field(s) and click on the button with single or double arrow pointing to the left (which ever is needed) and Click **Next** at the end.

- Step 4:** If you want to add a sub-form that displays data from another table that is related by a primary key, set a tick in the Add Sub form command. Click **Next**.
- Step 5:** Arrange Controls - In this step you can select the label text alignment, as well as the control(label plus data control) layout. Select fields layout by selecting one of the four offered possibilities, Click **Next**.
- Step 6:** Set Data Entry - There are two options in Set data entry. The first option is used to enter new data - existing data will not be displayed. Option two displays all data, but also gives the option to add restrictions. Select the type of data input. Click **Next**.
- Step 7:** Apply Styles - In this step you can select the look and feel of the form and controls. Select layout style of your form. Click **Next**.
- Step 8:** Set Name - Finally, you can set the name of the form, and also an option to continue to Work with form or to Edit the form by selecting the option **Modify the form**, then Click **Finish** button.

Modifying Form Design : The Forms created with wizard have limited visual appeal. However, Forms have a design view, just as table do, and LibreOffice base includes many tools for modifying a Form's design. Some of the common modifications to the Form are listed below :

- ◆ Changing Properties of controls
- ◆ Re-sizing and moving controls
- ◆ Aligning and spacing controls
- ◆ Converting (or Morphing) controls
- ◆ Conditional formatting of controls
- ◆ Re-arranging Tab Order
- ◆ Adding New controls
- ◆ Deleting existing controls

(b) Create Form by using Design View : Select '**Forms**' from the Database pane and click on '**Create Form in Design View...**' from the Tasks area

To bind the form to a particular table, Select the Forms Navigator button on the bottom strip menu. In the **Forms Navigator** dialogue box, right-click on **Forms** and select **New > Form**. Then right-click on **Form** (*not* Forms) and Select properties. After selecting the **Data** tab, select the required table from the '**Content**' pull-down menu. Close the dialogue box

After linking a table, select the Fields from that table in the form . To do this , Select the **Add Field**

button on the bottom strip menu. From the **Add field** dialogue box, drag the fields to where you want it in the form or simply double click on it then Close the dialogue box.

Then you should arrange the controls according to the layout you designed earlier. This may require moving some controls, adjusting some of their sizes, separating parts of the form with lines or boxes, selecting the background, etc. After completing all task save the form by giving the name of the form.

Adding a Title : The Form must be suitably titled for its identity, which should be self-descriptive. To add a title, click on **Label field** button from **Form Controls** tool bar. Then place the label field in the form, double click on label to display Properties dialog box and enter Title of form and make necessary formatting with the help different option in the properties dialog box.

Changing the Properties of Forms and Controls : The properties of forms and controls have been classified into three Tabs: **General, Data and Events**. It is not essential to know every available property to work well in designing Forms in LibreOffice Base. But it is always good idea to check up the property values if the object is not behaving the way it is expected to. To view the properties of an object or control, right click at the control and select the **Control** in the menu. The Form's property sheet can be opened by click **Form** button in **Form Controls** tool bar

Moving and Resizing controls : In order to move a control, first select it by a click action, then move the pointer to the edge of the selected control, ensuring that any of the re-sizing handles appearing as bold dot is not pointed at directly. The pointer turns its shape to a small hand. At this stage, hold mouse button pressed and drag the control to its new location. Movement of control beyond the bottom or right edge of the Form, leads to increasing the Form area automatically. also allows for combining of select and move step thereby making it easier and more efficient to reposition the control. A control can be re-sized by dragging the re-sizing handles at the corners and sides of the object. A change in the size of text control, however, does not result in changing the size of its underlying field because the size of the field is specified in table's design and can be changed only by modifying the properties of the field in table design.

Re-arranging the Tab Order : The Tab order of the Form (defined as a sequence of controls to move through when pressing a tab) is assigned while creating a Form. The tab order goes out of sequence when the controls in the Form are re-arranged. To change the tab order, click on **Activation Order** button in Form Design Tool bar. Clicking **Automatic sort** generally rearranges the fields in the correct order. It is preferable to try this option first. If the auto order is not correct, the tab order can be set manually by clicking the **Move Up** and **Move Down** in **Tab order** window. You can also define the index of a control through its specific properties by entering the desired value under Tab order in the **Properties** dialog of the control.

15.4 Information Using Queries

Accounting information that is presented in an accounting report is generated by creating and executing various queries using DBMS. The basics of creating such queries in LibreOffice Base have been described below.

15.4.1 Basics of Creating Queries in LibreOffice Base

Recall that one of the great advantages of relational databases is that the fragmented data is stored in different data tables so that there is no or minimum redundancy. But a complete view of data stored

across various tables is achieved only by executing queries based on SQL. A query is capable of displaying records containing fields from across a number of data tables.

15.4.2 Types of Queries

There are several types of queries in LibreOffice Base that are used to generate information. Such queries are called select queries because they are used to “select” records with a given set of fields: actual and computed and also for a given criteria. There are three important query types that are required for generating the accounting reports. These queries have been discussed as below:

(a) Simple Query : A select query is a simple query if it does not involve use of any query function to produce a summary of data. The criteria, if any, used in such a query is based on some constant value or values, forming an integral part of the query. For example, a query, to find date and amount of transactions records in which an account, identified by code = '711001' is debited, is a simple query and is executed, using database design of Model-I by the following SQL statement :

```
SELECT vDate, Amount
FROM Vouchers
WHERE Debit = '711001'
```

In the above SQL statement, the SELECT statement is meant to specify the fields to be selected, FROM clause specifies the source of data and WHERE clause filters the records matching the condition that Debit field has code = '711001'

(b) Parameter Queries : A parameter query prompts the user to enter parameters, or criteria through an input box, for selecting a set of records. A parameter query is useful when there is a need to repeat the same query with different criteria. The criteria, this means, is not constant as in the case of the simple query. While extracting the transactions to prepare ledger accounts, the same set of queries need be executed for different account codes. Consider the following SQL statement :

```
PARAMETERS AccountNo Text (255)
SELECT Name
FROM Accounts
WHERE Code = AccountNo
```

In the above query, the PARAMETERS clause is meant to declare the variable AccountNo. This SQL statement, when executed, prompts the user to provide the value of AccountNo.

(c) Summary Queries : A summary query, as opposed to a simple query, is used to extract aggregate of data items for a group of records rather than a detailed set of records. This query type is of particular importance in accounting because the accounting reports are based on summarisation of transaction data. Consider the following SQL statement :

```
SELECT Code, Name, Sum(Amount)
From Vouchers INNER JOIN Accounts
```

ON (Accounts.Code=Vouchers.Debit)

GROUP BY Code, Name

In the above query, the Vouchers table has been joined with Accounts table on the basis of Code field of Accounts and Debit field of Vouchers. The resultant record set has been grouped on the basis of Code and name of accounts. Accordingly, the sum of amount for each group (or set of records) has been ascertained and displayed. Finding the sum is the process of summarisation.

15.4.3 Adding Computed fields

The computed fields, representing secondary data, do not form part of data stored in tables because such data items unnecessarily increase the size of database. The secondary data items can always be generated on the basis of primary (or stored) data. In order to find values of such secondary data items, the query is based on computed fields. The computed fields provide up-to-date calculated results because they rely upon updated stored data values. For example, a data table, named Sales, which includes ItemCode, Quantity, Price, Dated and CustId, is maintained in a database to store sales transactions. In order to get list of sales transactions along with total sales relating to CustId='A051', the following simple query is executed by including Sales as computed field :

```
SELECT Dated, ItemCode, Quantity*Price AS Sales
```

```
FROM Sales WHERE CustId= 'A051';
```

In the above query the expression Quantity*Price has been given the name Sales by using AS clause.

15.4.4 Using Functions in Queries

A function in the Base environment is named and followed by parenthesis (). The function receives some inputs as its arguments and returns a value (also called its output). These functions also form a part of the expression for a computed field. Some commonly used functions have been described and discussed in Appendix given at the end of the chapter.

15.4.5 Methods of Creating Query

There are three ways in which any of the above queries can be created in LibreOffice Base. These methods are Wizard, Design View and SQL View. A brief description of each is given below :

(a) Wizard Method : In order to create a query using Wizard, the following steps are required. :

- (i) Select Queries from Objects list given in LHS (Left Hand Side) of Database window.
- (ii) Double click at Create Query by Using Wizard given on the RHS (Right Hand Side). Immediately, there is a window titled 'Query Wizard' (Shown in figure: 14.6) that prompts the user to select a field from a table or an existing query that is to be included in the query being created. Many such fields may be selected according to the information requirement of the query. The tables (or queries) being chosen represent the data source of the query being created. The fields being selected imply the data items to be displayed by the query.

- In the figure (Figure 15.6) the table 'emp' is selected. All the fields of the selected table

can be seen in the **Available fields** list. The user can select the field(s) needed from the list using the tools arranged (Like >, >>, < and <<) right to **Available fields** list. The selected fields can be seen in **Fields in the Query** list. The order of the selected fields can change using the tools (^ and v) right to **Fields in the Query** list. Then, press **Next** button or **Finish** button.

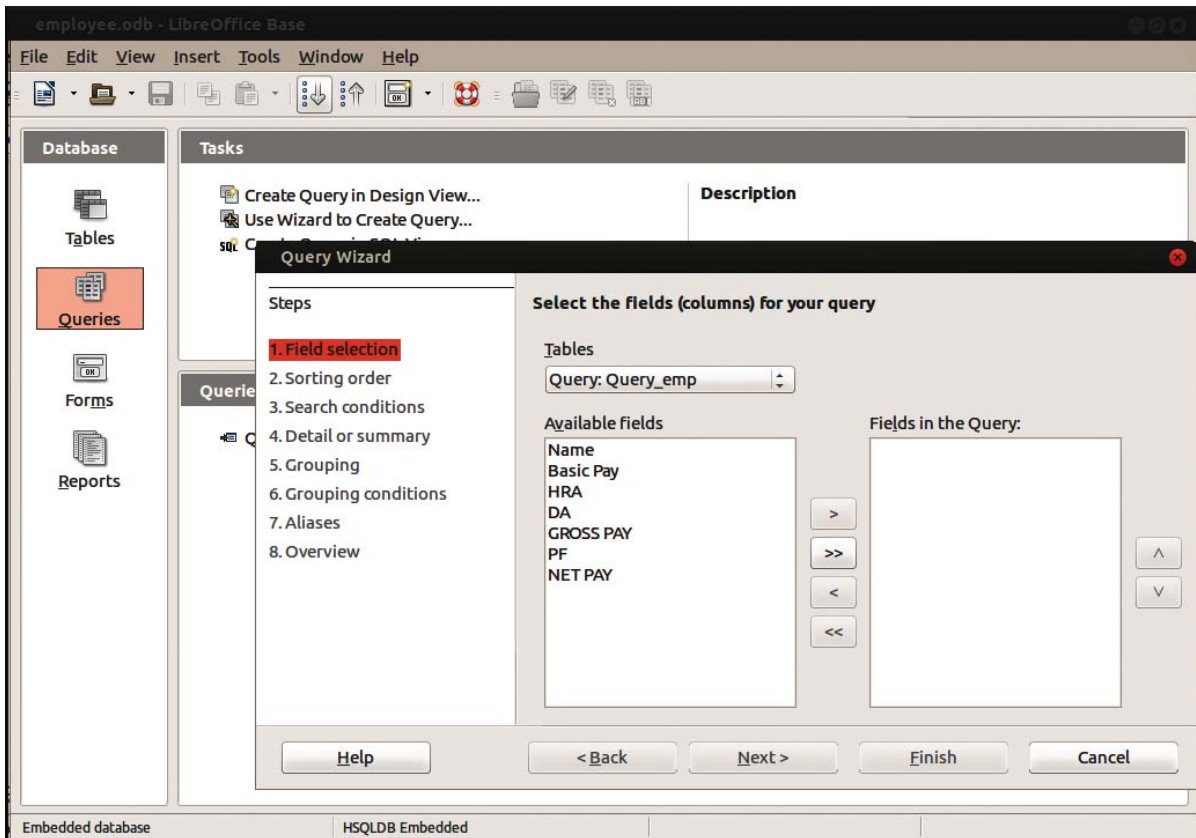


Figure 15.19 : Window to display simple query wizard

- **Sorting order:** In this step, the field name to sort the query result can be selected. In the given example (Figure 6.x), the basic pay (*bp*) is selected as **sorted by** field. After selecting the order (Ascending / Descending), press **Next** button. [Skip this step, if no sorting is needed].
- **Search Conditions:** This step specifies the search conditions to filter the query. The conditions can be stated using logical AND or logical OR (Figure 15.20). After stating conditions, press **Next** button. [This can skip, if no filtering is needed]
- **Details of Summary:** This page specifies whether to display all records of the query, or only the results of aggregate functions. This page is only displayed when there are numerical fields in the query that allow the use of aggregate functions.
- **Grouping:** Specifies whether to group the query. The data source must support the SQL statement 'Order by clauses' to enable this page of the Wizard.

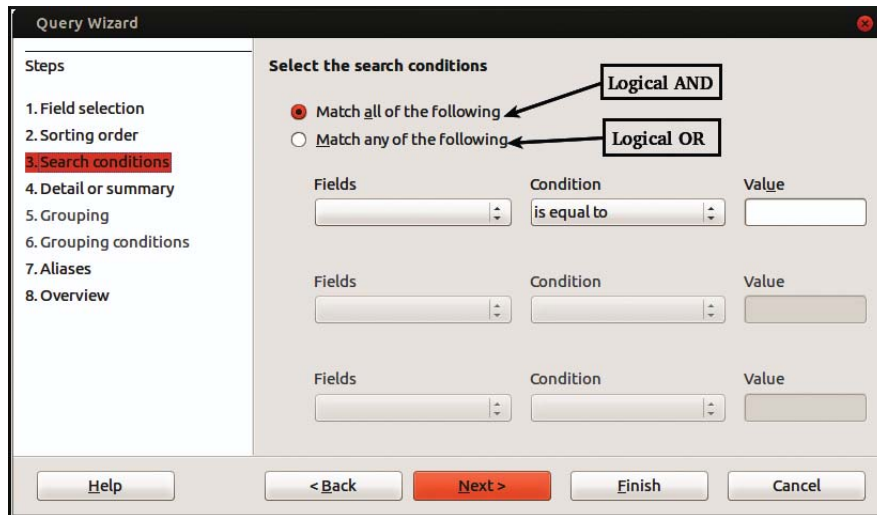


Figure 15.20 : Query Wizard Window to state conditions to filter the data records

- **Grouping Conditions:** Specifies the conditions to group the query. The data source must support the SQL statement 'Order by clauses' to enable this page of the Wizard.
- **Aliases :** This page helps to assign aliases to field names (Figure 15.21). Aliases are optional, and can provide more user-friendly names, which are displayed in place of field names. For example, an alias can be used when fields from different tables have the same name.

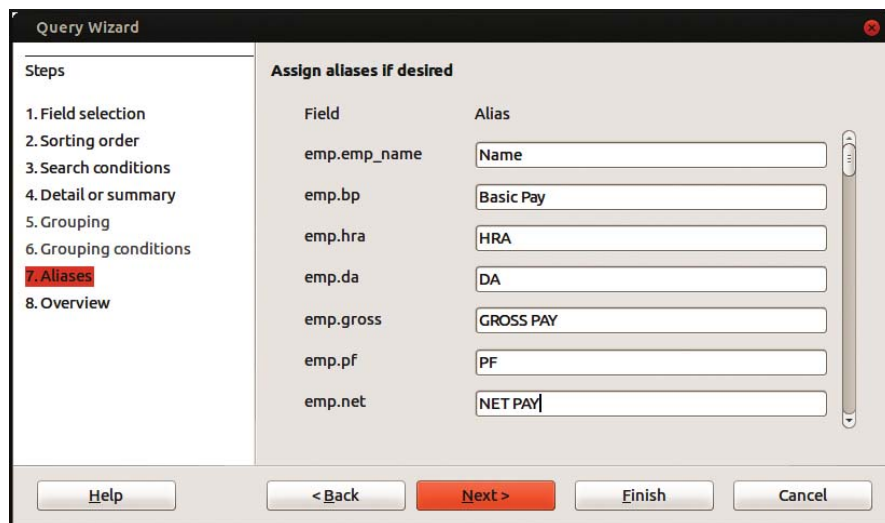


Figure 15.21 : Query Wizard - After assigning aliases to field names

- **Overview:** This wizard page gives an overview of the query made. It helps to enter a name of the query, and specify whether to display or to modify the query after the Wizard is finished.
- Press **Finish** button after the completion of Query wizard entry. The query will be saved. The user can run this query at any time. See the query result in Figure 6.x. Note the replacement of field names with aliases.

Query_emp - employee - LibreOffice Base: Table Data View							
File Edit View Insert Tools Window Help							
	Name	Basic Pay	HRA	DA	GROSS PAY	PF	NET PAY
	ANAND	35000	1500	4200	40700	6000	34700
	BENOJ	32000	1500	3840	37340	8000	29340
	SHENOY	29000	1500	3480	33980	5000	28980
	PRANAV	25000	1500	3000	29500	5000	24500

Figure 15.22 : Query Result

(b) Design Method : In order to create a query by design method, the following steps are required :

- Use the option **Create Query in Design View** from Base window (See Figure 15.19) to create query in design view. **Query Design** window appears (Figure 15.23).

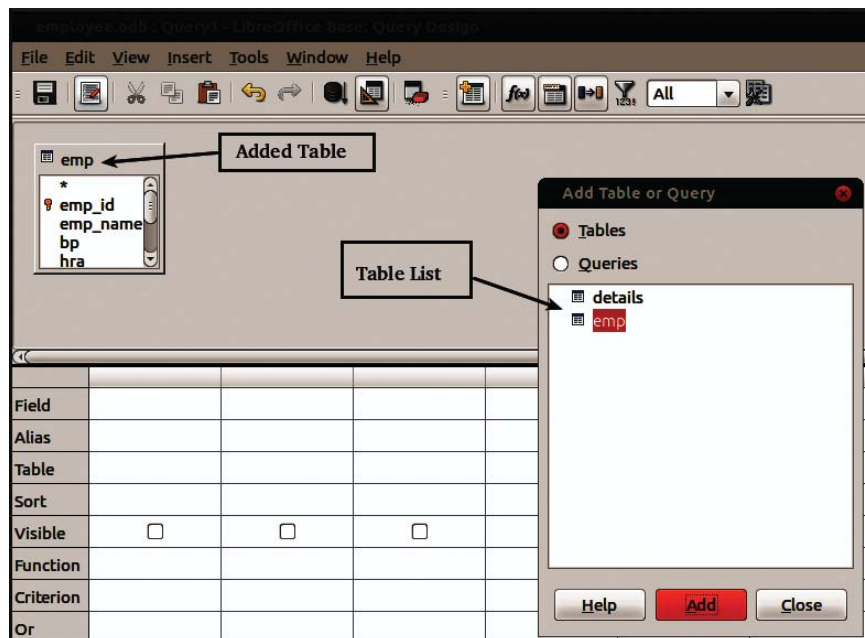


Figure 15.23 : Query Design Window

- Use **Add Table or Query** dialogue to include tables to query design. See added table **emp** in Figure 15.23. Then, include the fields and formula in the top row as shown in Figure 15.24. Give aliases in the second row, if needed. Press, **Run Query** button to execute the query. The result will be displayed on the top of query design table.

(c) SQL View Method : It can be done using the option **Create Query in SQL view** after selecting **Queries** from Database pane. Here, the SQL statements can be written by using keyboard. The desired SQL statement is directly saved in the same manner as described for design method. While forming the SQL statement, the following clauses are normally used for generating information (or Select) queries :

(i) **SELECT** : This clause is used to specify the fields to display data or information. Consider the following SQL statement segment :

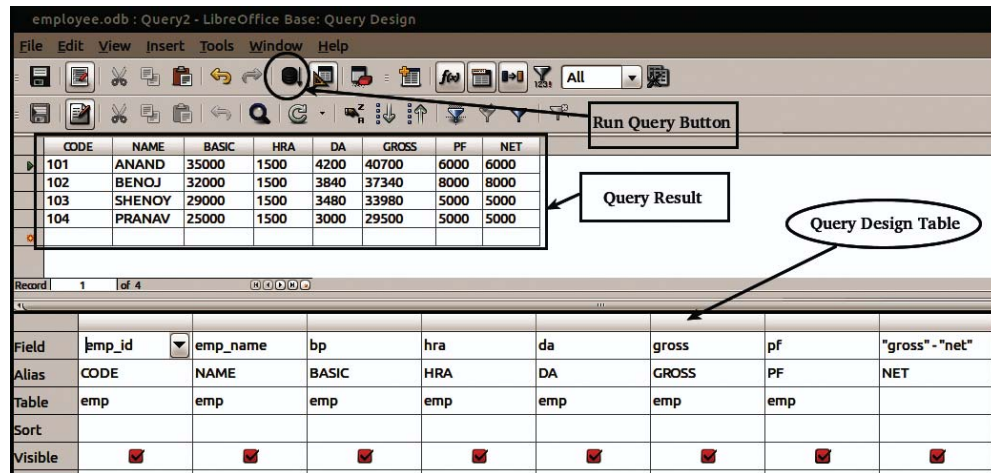


Figure 15.24 : Query Design Window after the execution of Query

SELECT Code, Name, Amount

The fields Code, Name and Amount after SELECT clause indicate the data items to be displayed by the query statement.

(ii) **FROM** : This clause is meant to indicate the source of data in terms of tables or queries or a combination of both. Two tables are joined by specifying a JOIN clause based on a condition of Join. There can be three types of Join: Inner, Left and right.

(iii) **INNER** : This Join clause is meant to display only exactly matching records between two data sources. Consider the following SQL statement segment:

FROM Accounts INNER JOIN AccountType

ON (CatId=Type)

In the above statement, only those records of Accounts and AccountType table constitute the source of query data, which match exactly on CatId = Type.

(iv) **LEFT** : With this Join, all the records in the primary table in the relationship are displayed irrespective whether there are matching records in the related table or not. Consider the following SQL statement segment :

FROM Accounts LEFT JOIN AccountType

ON (CatId=Type)

In the above statement, all records of Accounts along with matching records of AccountType table constitute the source of query data, The matching condition is CatId = Type.

(v) **RIGHT** : With this Join, all the records of related table in the relationship are displayed irrespective whether there are matching records in the primary table or not. Consider the following SQL statement segment

FROM Accounts RIGHT JOIN AccountType

ON (CatId=Type)

In the above statement, all records of AccountType along with matching records of Accounts table constitute the source of query data. The matching condition is CatId=Type.

(iv) **WHERE** : This clause in SQL statement is used to provide the condition to restrict the records to be returned by query. The resultant records of query must satisfy the condition which is specified after WHERE clause. This is meant to filter records returned by the query.

(v) **ORDER BY** : This clause is meant to specify the order in which the resultant records of query are required to appear. The basis of ordering is determined by the list of fields specified after the order by clause. Consider the following SQL statement segment :

ORDER BY Type, Code

The above statement in the context of Accounts table implies that the resultant record set is ordered by the Type field of Accounts and within Type, by Code field of Accounts.

(vi) **GROUP BY** : The group by clause is used in the SQL statement to enable grouping of records for creating summary query. The fields after GROUP BY clause constitute the basis of grouping for which summary results are obtained. Consider the following SQL statement:

SELECT Debit, Sum(Amount)

FROM Vouchers

GROUP BY Debit

In the above SQL statement, the GROUP BY clause uses Debit account codes as the basis for computing the sum of amount of voucher. The total amount, by which every transacted account has been debited, is given by this SQL statement In this case, sum of amount is found for each group of records formed using GROUP BY clause.

Generating Accounting Reports

An Accounting system without reporting capability is incomplete. The output of accounting system takes the form of accounting reports. LibreOffice Base offers a great flexibility in designing and generating customised reports.

15.5.1 Accounting Reports

Every report consists of 'information', which is different from 'data'. Data processing leads to data transformation and when this processing is in accordance with decision usefulness, it is called information. Information generation is the process of compiling, arranging, formatting and presenting information to the users.

A report is prepared with a definite objective. Every report is a collection of related information for a particular need and purpose and must meet the twin objectives of reporting : one to reduce the level of uncertainty that is faced by a decision-maker; second to influence the behaviour (or positive actions) of the decision-maker. Accordingly, accounting information, generated by processing accounting data is gathered to generate an accounting report. An accounting report, therefore, is the physical form of accounting information. Useful accounting information, regardless of its physical form, must have five characteristics: relevance, timeliness, accuracy, completeness and summarisation.

An accounting report, in order to be useful, must display information content in such a manner as to give confidence to the user, influence his behaviour and prompt him to take positive actions. Reports, which do not meet the above stated objectives, have no value. There are two broad classes of accounting reports: Programmed and Casual (also called Adhoc or Pass through).

(a) Programmed Reports :

These reports contain information useful for decision-making situations that the users have anticipated to occur. There are two types of reports within this report type: Scheduled and On demand.

- ◆ **Scheduled Reports :** The reports, which are produced according to a given time frame, are called scheduled reports. The time frame may be daily, weekly, monthly, quarterly or yearly. Some examples of scheduled reports are: Trial Balance, Ledger, Statement of Cash Transactions (Cash Book), Closing Stock Report, Profit and Loss Account and Balance Sheet, etc.
- ◆ **On Demand Reports :** The reports, which are generated only on the triggering of some event, are called On demand reports. Some examples of On demand reports are a Customer's Statement of Account, Inventory Re-order Report, Stock in hand Report for a Selected Group of items, etc.

(b) Casual Reports :

There are reports, the need for which is not anticipated, the information content of which may be useful but casually required. These are adhoc reports and are generated casually by executing some simple queries without requiring much of professional assistance. As opposed to programmed reports, casual reports are generated as and when required.

15.5.2 Process of Creating Reports

The process of generating accounting reports in LibreOffice Base involves three steps: designing the report, identifying the accounting information queries, and finally creating an accounting report by using such queries.

(i) Designing the Report :

Every report is expected to meet certain objectives of reporting for which it is designed and developed. It should not be too big or too small. Objective-oriented reporting means designing the report in such a manner as to meet the pre-conceived objectives in view.

(ii) Identifying Accounting Information Queries :

A number of SQL statements are written and refines its results by using fresh data (or information) from existing data tables (or queries).

(iii) Using the Record set of Final SQL :

The record set of final SQL that relies upon preceding SQL statement, is collection of report-oriented information. This record set need be embedded in the report being produced.

15.5.3 Basics of Designing a Report in LibreOffice Base:

A report is a presentation of stored or transformed data in an organised manner. LibreOffice Base

saves the design of the report, which consists of information structure along with various controls to display information content and its record source. When a saved report is opened, the information content is retrieved from the tables and displayed according to the design. There are two types of formats of presenting information through a report: Columnar and Tabular.

- ***Columnar Report Format :***

A columnar format displays the caption of each field on a separate line in a single column down the page. The corresponding information contents of the fields are shown in another column next to their respective fields.

- ***Tabular Report Format :***

A tabular format displays the caption of fields on the same line so that their respective information contents appear in the next line. The number of columns in tabular report is exactly equal to the number of fields to be displayed.

15.5.4 Structure of Report in LibreOffice Base

A report is designed using three major sections which taken together constitutes the structure of report design. It is not necessary that every report designed must have all the sections that have been described below:

- ◆ ***Page Header*** : Page header appears at the top of every page of the report. It may include a uniform title to indicate that the page belongs to a particular report.
- ◆ ***Details*** : The details section, which is also called the main body of a report, contains data from tables or queries that provide the record source to a report. This section is most important as it consists of the main information content of a report.
- ◆ ***Page Footer*** : The page footer appears at the bottom of each page of the report and is meant to include page numbers, date and time of report generation.

15.5.5 Methods of Creating a Report

There are two ways in which a report can be created in LibreOffice Base. A brief description of each method is given below:

(a) Wizard :

The Report wizard allows a designer to choose the fields from multiple tables along with specification for grouping, sorting and formatting of information content in report. In order to create reports by wizard, following steps are required.

- (i) After selecting **Reports** object, double click at **Create Report by Using Wizard**.
- (ii) Choose the Field Selection for which field do you have in your report that includes information content of report from Report wizard on left pane.
- (iii) Use arrow buttons to select fields to provide the information source to report. Single right arrow button is used to select one field and double arrow button to select all fields. Alterna-

tively, double click at the fields to be selected in the same order in which they are required to be displayed in the report.

- (iv) Choose the Labeling Field to change the label as you wish.
- (v) Add any grouping level(s) for displaying the content of the report, if any.
- (vi) Specify the sort order based on any of the fields contained in the report. The records may be sorted by specifying either ascending or descending order for each field.
- (vii) Choose the report layout (Tabular, columnar single-column, columnar two columns, columnar three columns, in blocks labels above, in blocks labels left) and its orientation (portrait and landscape). Click Next after specifying the layout and orientation.
- (viii) Click on the option **Create report** enter the report name, select style of report (static report and dynamic report), and choose any of the options- modifying report layout and create report now. Click Finish button for report generation.

(b) Design View :

The design view method offers greatest flexibility to the designer in designing a report. In this method, the report is designed by assembling and embedding various components from report tool box. In order to design a report by using design view, following steps are required:

- i) After selecting Reports object, double click Create report in Design view. LibreOffice Base responds by providing a blank report object with three sections: Report/Page header, Detail and Report/Page footer as shown in Figure : 15.25.

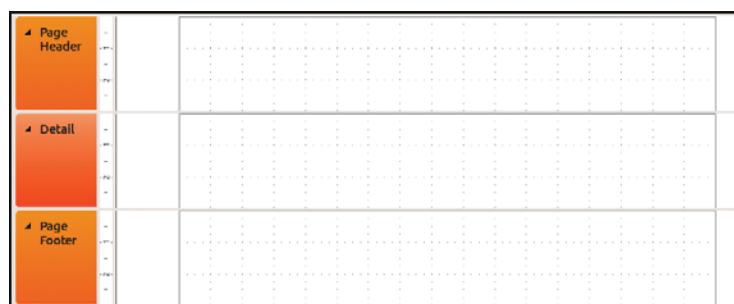


Figure 15.25 : Report Builder

- (ii) Right click the mouse at the black spot appearing at the left of horizontal ruler of above report. Report object responds by displaying a drop down window.
- (iii) Click Properties and select Record Source from Data tab. The record source turns into a combo control giving a list of various tables and queries. Choose the appropriate source of information to be presented in the report being designed. LibreOffice Base responds by providing a list of fields of the selected record source. If this list does not appear or it is closed by mistake, it can be recalled by clicking at the field list icon appearing before the icon for tool box.
- (iv) Select the required fields from list of fields displayed as discussed in (iii) above, by clicking at each of the fields to be selected while keeping the Ctrl key pressed. Drag and drop the selected fields to Detail section.

- (v) The label part of each field is moved to Report/Page header and text part is accordingly aligned below their respective labels column wise. The caption of each label giving headings can be suitably modified, if required.
- (vii) The vertical ruler controlling the distance between various report sections can be suitably adjusted to give a better look to the report. The Report/Page footer bar is brought close to the fields laid out in Detail section so that the gap between records of details section is minimized.
- (viii) Page headers and page footers may also be added by right click at title bar of report object, followed by click at Page header/footer.
- (ix) then click **execute the report** from the tool bar for create the report

15.5.6 Refining the Report Design:

The design of the report created by any of the methods described above may be improved upon by making the following additions and modifications to the report. For this purpose, an existing report is opened in design mode.

Adding Dates and Page Numbers :

When an existing report is opened in design mode, the page footer of the report contains two unbound controls: the current date and current page number of total number of pages. Both the controls may be customised according to the requirement of the designer. The date control uses = Now() function to retrieve the current date. The format of date may be modified by selecting General date, Medium date, Short date, Long date etc. from format property of this control. Further, when a report is created using design view method, the date and/or time and also the page numbers may be added to any of its part. The date and time is added by clicking Insert date and time from the menu bar to open the Date and Time dialog box. After selecting and specifying the desired preferences regarding date and time, click OK to find that a text control with chosen date and time preferences is added at the top of active report section. This added text control containing date and time may be dragged and dropped in any part of the report as per requirement. Similarly, the page number is added by clicking Insert page numbers from the menu bar to open the Page numbers dialogue box. This dialogue allows the designer to specify the format, position and alignment. The two formats are: Page N (for example Page 1) and Page N of M (for example Page 1 of 10). The position to specify is either Top of Page (header) or Bottom of Page (footer). Possible alignment, which may be specified are Centre, left, right, inside and outside.

Adding and Deleting Report Controls :

After a report has been designed, additional report controls may be added or deleted by the same procedure as applicable to forms. Clicking tool bar icon opens report design tool bar, which contains a set of useful controls.

- (a) After opening the report in design mode, click Field List button on report design tool bar. This results in opening the field list window.
- (b) Drag the field into an appropriate section of the report. The field appears with both label and text box control. The label part gives a constant field heading while the text part provides different values of the field. These two parts are accordingly placed at the appropriate sections of the

report.

- (c) A field control may be deleted by selecting the control and pressing the Delete key.

Conditionally Formatting Report Controls :

The conditional formatting of text boxes and combo boxes in reports can be achieved in the same manner, as it applies to Forms. The conditional formatting allows the designer to apply special text formats that depend on the value of field. This facility is a useful tool to draw the attention of user or reader of report to some values of particular interest, such as amounts exceeding certain limit or unexpected balances in some accounts. In order to create a conditional formatting, following steps are required:

- (a) Open the report in design view.
- (b) Select a control and click at format on menu bar, followed by conditional formatting.
- (c) Provide the necessary conditions for formatting to occur in the same manner as already discussed while applying conditional formatting to design of Forms.
- (d) The conditional formatting is removed by re-opening the same dialog and clicking at delete button.

Grouping Levels and Sorting Order :

The purpose of grouping is to organise the information content of a report into categories. Sorting order is meant to arrange such information content into numerical or alphabetical order. With groupings the sorting applies to each individual group. The grouping and sorting of information, when applied together, make the report more meaningful and therefore useful to the user of the report. In order to specify the grouping and sorting order, following procedure is adopted.

- (i) Click at Sorting and Grouping icon of Report Design Tool bar (This icon is located next to icon for tool box). Immediately, LibreOffice Base responds by displaying the following Sorting and Grouping dialogue box.
- (ii) The left pane of this dialog box provides a list of fields or expressions that are to be used for grouping and sorting. In the above dialog box, Type field of Accounts has been chosen as the basis of grouping the information content of trial balance. The group header and footer property is set to Yes to indicate that there is separate header and footer for each group of accounts in trial balance.

15.5.7 Saving and Exporting a Report

After a report is designed, it may be generated to preview its final shape. Both the design and a generated report are saved for future use and reference. The generated report may also be exported for use by others, as described below:

- (a) Saving and Exporting Report Object in LibreOffice Base : The design of a report is saved in LibreOffice Base as report object by assigning a particular name. The report object, when opened in LibreOffice Base by click action generates the desired report as per design specification. The design may also be exported to another database file of LibreOffice Base. This is achieved by

clicking File Export and then selecting an existing database into which the report design is to be exported. LibreOffice Base responds by providing a dialog box to give the name by which the exported report is saved in a selected database.

- (b) Exporting to LibreOffice Calc: A generated report may be exported to Calc, which is a spreadsheet package.
- (c) Exporting to LibreOffice Writer LibreOffice Writer : A report generated using LibreOffice Base can also be exported to LibreOffice Writer
- (d) Printing a Report : A generated report may also be printed by taking the following steps provided a printer attached to the computer is installed.
 - (i) Choose File from menu bar Print
 - (ii) LibreOffice Base responds by providing a print window, which allows the user to select a printer, the number of copies to be printed and also the range of pages to be printed.
 - (iii) Properties button is clicked to define print quality under set-up tab and orientation under paper tab. Two-sided printing may also be obtained if the printer supports this feature.
- (e) E-Mailing a Report : A report generated by LibreOffice Base may also be sent using E-Mail facility, provided the computer system has Internet facility and is connected to the Mail Server of the Internet Service Provider (ISP). In order to send a report using E-mail facility, following steps are required :
 - (i) Select and generate a report in Database Window
 - (ii) Click at File → Send-To → Mail recipient from Menu bar of LibreOffice Base. A Send dialog box appears with various options for choosing the Format:
 - (iii) Choose an appropriate format and click OK. LibreOffice Base responds by providing an E-Mail composition window.
 - (iv) Fill up the details regarding E-Mail address of recipient and others to whom copy of report is to be sent; provide a subject to E-Mail and click at Send button. The report gets dispatched to the mailbox of the recipient of E-Mail.

15.5.8 Designing Accounting Reports using LibreOffice Base

Financial Accounting Reports such as Cash book, Bank book, Ledger Accounts and Trial Balance may be generated in LibreOffice Base by adhering to report generation process. The exact process in the context of each of these reports is described below :

Trial Balance:

The Trial Balance is one of the accounting reports, which provides the net amount by which each account, during a given period of time, has been debited or credited. The format of a typical trial balance is as given below :

Trial Balance

Account Title	L.F.	Debit Amount (Rs.)	Credit Amount (Rs.)
Total			

Table 15.2 : Format of trial balance

To produce a trial balance, it is necessary to retrieve a set of processed data records each of which provides information on Code (or Account Number), Name of Account (or Particulars), Debit balance and Credit balance with reference to a each account. In order to find net balance corresponding to every account along with its identity, following steps are taken :

- (i) To find the total amount by which every account has been debited;
- (ii) To find the total amount by which every account has been Credited;
- (iii) To find a collective record set of accounts with their debit and credit totals;
- (iv) To find the net amount with which every account has been debited or credited; and
- (vi) To find the record set which consists of Account code, name of Account, Debit and Credit Amount.

Above steps to produce trial balance are transformed into a series of SQL statements, which vary according to the database design. The details of the above procedure along with the relevant SQL statements need be explained in the context of the three Models as given below :

Model-I : The following series of SQL statements retrieve a record set for producing trial balance when database design for Model-I is used.

- (a) To find the total amount by which the accounts have been debited : In order to ascertain the total amount by which every transacted account has been debited, the SELECT clause need to have two fields: one code to identify the transacted account and another to generate the total by which such account has been debited. This is achieved by using Debit field of Vouchers table and finding the sum of amount corresponding to each of the transacted accounts. The FROM clause relies upon Vouchers table to get the data source. The GROUP BY clause specifies the field on the basis of which grouping of record set is formed. This grouping is necessary in SQL when aggregate query is used to generate summary information. The summing of amount is obtained by using aggregate function, Sum(). This function, as already explained, uses a field with data type Number, as an input argument and returns its sum as output. Accordingly, the following SQL statement is formed :

SELECT Debit AS Code, Sum(amount) AS Total FROM vouchers GROUP BY debit; In the above SQL statement, the GROUP BY clause retrieves the rows of vouchers table accounts-wise because the debit field refers to account code. As a result, the Sum() computes the sum of amount of

a particular debit account and reports against Debit account of SELECT clause. This SQL statement is saved as Query 01 for its subsequent use. The total of debit amount in this query is given by Total field with positive amounts.

- (b) To find the total amount by which the accounts have been credited : In order to ascertain the total amount by which every transacted account has been credited, a query similar to that in (a) need be formed, except that the Debit field in SELECT and GROUP BY clause is substituted by Credit field. The sum of amount generated by sum(Amount) is multiplied by – 1 so that the final amount assigned to Total field is always negative. This is because the amount of credit must be a negative amount if amount of debit is taken as positive. The purpose of using negative values is to differentiate between debit and credit totals for each account and also to facilitate the simple arithmetic summation for obtaining the net amount.

Accordingly, the following SQL statement is formed :

```
SELECT Credit AS Code, Sum(Amount)*(-1) AS Total
```

```
FROM vouchers
```

```
GROUP BY Credit;
```

This SQL statement is saved as Query 02 to be used as source by next query.

- (c) To generate a collective record set of accounts with their debit and credit totals : Every transacted account that has been debited (or credited) only appears once in this collective record set. However, those transacted accounts that have been debited as well as credited appear twice in this record set: once with a positive amount and thereafter with a negative amount. This collective record set is generated by executing a UNION query between Query 01 and Query 02.

```
SELECT*
```

```
FROM Query 01
```

```
UNION SELECT*
```

```
FROM Query 02 ;
```

This SQL statement is saved as Query 03 for further processing of its resultant record set.

- (d) To generate the net amount with which an account has been debited or credited : Once the records of account codes with debit and/or credit totals have been collected, the next logical step is to find out the net amount by which such accounts have been either debited or credited. This is accomplished by forming another aggregate query in which FROM clause uses Query 03 as the data source. The sum of Total for each Code of data source, provided by Query 03, results in computing net amount for every account. Accordingly, the following SQL statement is formed to generate a list of account codes with their respective balances: positive or negative.

```
SELECT Code, Sum(Total) AS Net
```

```
FROM Query 03
```

```
GROUP BY Code;
```

A positive net amount implies a debit and negative amount means a credit balance corresponding to an account code. This is because in Query 02, the total of credit amount has been made to appear as negative. This query is saved as Query 04 for its subsequent use in generating record set for trial balance.

- (e) To find that record set which consists of account code, name of account, debit amount and credit amount : Every row of a trial balance report consists of Account Code, Name of Account, Debit Amount and Credit Amount. The Debit Amount and Credit Amount are mutually exclusive. Such rows are obtained by generating a record set based on the following SQL statement.

```
SELECT a.Code, b.name AS [Name of Account], IIF
```

```
(a.Net>0,a.Net,null) AS Debit,
```

```
IIF (a.Net<0,abs(a.Net) ,null) AS Credit
```

```
FROM Query 04 AS a, Accounts AS b
```

```
WHERE a.code = b.code ;
```

In the above SQL statement, the results of Query 04 and data stored in Accounts table has been used. The SELECT clause of this SQL statement has two computed fields as explained below :

- ◆ IIF(a.Net>0,a.Net,null) AS Debit: According to IIF() function, if the net amount exceeds zero, it is displayed as Debit, otherwise nothing appears in Debit field.
- ◆ IIF(a.Net<0,abs(a.Net) ,null) AS Credit: According to IIF() function, if the net amount is less than zero (implying negative), it is displayed as Credit, otherwise nothing appears in Credit field.

Besides, the other two fields: Code and Name, of SELECT clause are retrieved from Query 04 and Accounts table respectively. This SQL statement is saved as Query 05 for providing the necessary information content for Trial Balance Report.

Model-II : The following series of SQL statements retrieve the record set for producing trial balance when database design for Model-II is used. In addition to this, the accounts have been categorised within the trial balance according to the Account Type: Expenses, Revenues, Assets and Liabilities.

- (a) To find the total amount by which the accounts have been debited : The transacted accounts in design of Model-II have been stored in AccCode of VouchersMain and Code of VouchersDetail. The following SQL statement is formed to generate the relevant information from VouchersDetails.

```
SELECT Code, Sum(amount) AS Total
```

```
FROM vouchersMain INNER JOIN vouchersDetails ON
```

```
VouchersMain.Vno = VouchersDetails.Vno
```

```
WHERE Type = 0
```

```
GROUP BY Code ;
```

Similarly, the following SQL statement is formed to generate the required information from VouchersMain table.

```
SELECT AccCode As Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 1
GROUP BY AccCode ;
```

Both the SQL statements are meant to extract similar sets of records, but from two different sources. Therefore, the resultant record set of these SQL statements have been horizontally merged using UNION clause as shown below:

```
SELECT Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 0
GROUP BY Code
UNION ALL
SELECT AccCode As Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 1
GROUP BY AccCode ;
```

The above SQL statement is saved as Query101 for its subsequent use. The total of debit amount in this query represents the Total with positive amounts.

- (b) To find the total amount by which the accounts have been credited : In order to ascertain the total amount by which every transacted account has been credited, a query similar to that in (a) need be formed. This is achieved by substituting Debit field in SELECT and GROUP BY clause by Credit field and the sum of amount generated by sum(Amount) is multiplied by -1 so that the final amount assigned to Total field is always negative. Accordingly, the following SQL statement is formed :

```
SELECT Code, sum(amount)*-1 AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno=VouchersDetails.Vno
```



```

WHERE Type=1 GROUP BY Code, Amount

UNION

SELECT AccCode As Code, sum(amount)*-1 AS Total

FROM vouchersMain INNER JOIN vouchersDetails ON

VouchersMain.Vno=VouchersDetails.Vno

WHERE Type=0 GROUP BY AccCode, Amount;

```

In the above SQL statement, the sum of amount has been multiplied by -1 to ensure that the amount of credit is always negative just as amount of debit is taken as positive. This query is saved as Query102 for its subsequent use.

- (c) To find a collective record set of accounts with their debit and credit totals:

A collective record set is generated by forming a union query between Query101 and Query102 to ensure that the debit and credit amount with respect to each account becomes available for generating the net amount. Accordingly, the following SQL statement is formed.

```

SELECT*

FROM Query101

UNION Select*

FROM Query102;

```

The above SQL statement causes horizontal merger of record sets returned by Query101 and Query102. This SQL Statement is saved as Query103 for its subsequent use in next query.

- (d) To find the net amount with which an account has been debited or credited:

To generate the net amount, an SQL statement similar to Query04 (designed for query (d) of Model-I) above, is formed as shown below, except that its source of data is Query103 instead of Query 03.

```

SELECT Code, Sum(Total) AS Net

FROM Query103

GROUP BY Code;

```

This query is saved as Query104 for its subsequent use in generating a record set, giving details of information for trial balance.

- (e) To find the record set which consists of Account code, Name of Account, Debit Amount and Credit Amount : This query, which is meant to provide relevant information to the trial balance report, is similar to Query 05 (designed and discussed in (e) of Model-I). Accordingly, the following SQL statement is formed by changing the source of data from Query 05 to Query105 as shown

below :

```
SELECT a.Code, b.name AS [Name of Account], IIF(a.Net>0,a.Net,null) AS
Debit, IIF(a.Net<0,abs(a.Net) ,null) AS Credit FROM Query104 AS a,
Accounts AS b/
WHERE a.code = b.code;
```

In above SQL statement, the results of Query104 and data stored in accounts table has been used. This SQL statement is saved as Query105 for providing source of information to Trial Balance Report.

Trial Balance with Sorting and Grouping levels : In order to prepare a trial balance with all the account duly grouped by and sorted within category of accounts, two additional queries (f) and (g) are required.

(f) To find the record set of accounts with their category and category ID :

Accounts table is related to AccountType table vide Type field. The following SQL statement, using INNER JOIN clause, is formed to retrieve the relevant fields for various accounts.

```
SELECT Accounts.Code, Accounts.Name, Category, CatId FROM Accounts
INNER JOIN AccountType ON
Accounts.Type = Account type.CatId;
```

This SQL statement is saved as Query 106 for its subsequent use in next query.

(g) To find the record set consisting of Account Code, Name of Account, Debit Amount and Credit Amount along with category details : This query, when compared with (e) above, reveals that two additional fields: Category and CatId are required. Accordingly, the SQL statement stored as Query105 is modified by substituting Accounts table with Query106 to form the following Statement.

```
SELECT a.Code, b.name AS [Name of Account],
IF(a.Net>0,a.Net,null) AS Debit, IIF(a.Net<0,abs(a.Net) ,null) AS Credit,
Category, CatId
FROM Query104 AS a, Query106 AS b
WHERE a.code = b.code ;
```

This SQL statement is saved as Query107 to provide information details for designing trial balance with grouping and sorting of the accounts. 15.5.9 Procedure in LibreOffice Base for Designing a Simple Trial Balance The Trial Balance is generate. Adding Dates and Page Numbers : When an existing report is opened in design mode, the page footer of the report contains two unbound controls: the current date and current page number of total number of pages. Both the controls may be customised

according to the requirement of the designer. The date control uses = Now() function to retrieve the current date from RTC of computer. The format of date may be modified by selecting General date, Medium date, Short date or Long date from format property of this control. Further, when a report is created using design view method, the date and/or time and also the page numbers may be added to any of its part. The date and time is added by clicking Insert % date and time from the menu bar to open the Date and Time dialog box. After selecting and specifying the desired preferences regarding date and time, click OK to find that a text control with chosen date and time preferences is added at the top of active report section. This added text control containing date and time may be dragged and dropped in any part of the report as per requirement. Similarly, the page number is added by clicking Insert % page numbers from the menu bar to open the Page numbers dialogue box. This dialogue allows the designer to specify the format, position and alignment. The two formats are: Page N (for example Page 1) and Page N of M (for example Page 1 of 10). The position to specify is either Top of Page (header) or Bottom of Page (footer). Possible alignment, which may be specified are centre, left, right, inside and outside.

- ◆ Adding and Deleting Report Controls : After a report has been designed, additional report controls may be added or deleted by the same procedure as applicable to forms. Clicking tool bar icon opens report design tool bar, which contains a set of useful controls.
- (a) After opening the report in design mode, click Field List button on report design tool bar. This results in opening the field list window.
- (b) Drag the field into an appropriate section of the report. The field appears with both label and text box control. The label part gives a constant field heading while the text part provides different values of the field. These two parts are accordingly placed at the appropriate sections of the report.
- (c) A field control may be deleted by selecting the control and pressing the Delete key.

Conditionally Formatting Report Controls : The conditional formatting of text boxes and combo boxes in reports can be achieved in the same manner, as it applies to Forms. The conditional formatting allows the designer to apply special text formats that depend on the value of field. This facility is a useful tool to draw the attention of user or reader of report to some values of particular interest, such as amounts exceeding certain limit or unexpected balances in some accounts. In order to create a conditional formatting, following steps are required:

- (a) Open the report in design view.
- (b) Select a control and click at format on menu bar, followed by conditional formatting.
- (c) Provide the necessary conditions for formatting to occur in the same manner as already discussed while applying conditional formatting to design of Forms.
- (d) The conditional formatting is removed by re-opening the same dialog and clicking at delete button.

ed using the Design View method by following the steps listed below :

- (i) Select Reports from objects list provided by LHS of Database Window and click at New object button of tool bar. LibreOffice Base responds by displaying the New Report Window as shown in figure 15.8 Choose Design View from list of methods and Query 05 from combo

control meant to provide data source to the report. Click OK after choosing method and data source of report.

- (ii) LibreOffice Base responds by displaying a blank report design divided horizontally into three sections: Page Header, Detail and Page Footers. Besides, a list of available fields of Query 05 is also provided for embedding on to this blank design of report.
- (iii) Alternatively, double click at Create report in design view. LibreOffice Base respond by displaying a blank report design duly divided into three sections as stated above. Right Click at the left most corner point of report design where horizontal and vertical rulers converge. Click at Properties of report and select Data tab to define the record source as Query 05. Immediately, there appears as list of available fields of Query 05 so as to be placed on to blank design of report. (iv) Right click at any part of the report design and choose Report Page Header and Footer. LibreOffice Base responds by providing two more sections:

Page Header and Page Footer.

- (v) Click at the icon for tool bar and pick up a label control to be placed at Page Header Section and assign set its caption property to Trial Balance, Font Size to 16, Font colour to Blue, Text align to Left and Font weight to Bold.
- (vi) Select all the fields of Query 05 by clicking at every field while keeping the Ctrl key pressed. Drag and drop the selected fields on Details section. It may be noted that each of the dropped fields has two controls: Label and Text. The former gives caption and the latter provides the data content.
- (vii) Select the label controls of all the four fields by clicking at each while keeping the Shift Key pressed. Right click at selected label controls and choose cut. Place the mouse at Page Header section and paste these controls.
- (viii) Re-arrange these label controls to appear as headings of columns for trial balance as: Code, Name of Account, Debit and Credit. Select all these label controls and right click to choose properties. LibreOffice Base provides Properties of these controls. Choose format tab and set the Font weight Property to Bold; Font Size to 10; Font colour to Blue and Text align to Centre.
- (ix) Align the Text controls in Detail section to appear just below each of the respective label controls appearing in Page Header section.
- (x) Select the Text controls and Debit and Credit field and modify their properties by setting Decimal Places to Zero and Format to Standard.
- (xi) Pick up a label control from tool box by click action and place at Report Footer section, at the area vertically below the column “Name of Accounts” and give the caption “Total”. Set its Text align property to Centre, Font weight property to Bold and Font Size to 10.
- (xii) Pick up a text control and place it at Report Footer section at the area vertically below Debit column. Set its Record source property as expression given below :

= Sum ([Query 05]![Debit])

The expression is written by clicking at (...) to call the expression pane.

The expression [Query 05]![Debit] within Sum() function refers to Debit field of Query 05.

- (xiii) Pick up another text control and place it at Report Footer section at the area vertically below Credit column. Set its Record source property as expression given below.

=Sum ([Query 05]![Credit])

The expression is written in the manner as it applies to sum of debit column. The expression [Query 05]![Credit] within Sum() function refers to Credit field of Query 05. The report design prepared above is saved as Trial Balance by Design. The Trial Balance report design appears on the RHS of Database Window as object under Reports.

15.5.10 Designing of Trial Balance with Sorting and Grouping

To design a trial balance with grouping and sorting of accounts, the following additional steps are required.

- (i) Copy the trial balance design as created above and paste it with different name say “Trial balance with Grouping”. Open this copied report design for modification in design view to incorporate the grouping and sorting of accounts in trial balance report.
- (ii) Change the data source property of report design by right click at the top left corner of report design % click at properties % Choose Tab and set the Record source property as Query107.
- (iii) Modify the Record source of Text controls for sum of debit and credit columns to replace existing expressions by

= Sum ([Query107]![Debit]) for Debit

= Sum ([Query107]![Credit]) for Credit

- (iv) Right click at report design % click at sorting and grouping. LibreOffice Base responds by providing a window for sorting and grouping as shown in figure 15.12
- (v) Define the basis of grouping as CatId in field/expression and its sort order set to ascending. Set the Group Header property to Yes. LibreOffice Base responds by inserting CatId Header section in report design.
- (vi) Click at field list icon and drag and drop category field in CatId Header section. Set its Font Size property to 10, Fore Colour property to Dark Green and Font Weight property as Bold.

Save the modifications in the above report design. The trial balance report is generated by double click at this or the previous object. The generated trial balance may be saved or exported as desired.

Questions for Practice

Short Answers

1. State what do you understand by accounting reports.
2. What do you mean by programmed or casual reports?
3. With the help of an example, briefly state the meaning of parameter queries.
4. Briefly state the purpose of functions in SQL environment.
5. Briefly explain in steps the method of creating a query, using wizard, in LibreOffice Base.
6. List the structure of a good report created in LibreOffice Base.
7. List the ways to refine the design of a report.
8. Briefly explain the purpose of grouping and sorting of the data as a means to refine a report.

Long Answers

1. Discuss the concept of accounting reports? Explain the steps involved in creating such reports.
2. Discuss with a set of inter-related data tables, the basics of creating queries in LibreOffice Base?
3. Describe in steps the design view method to create a query in LibreOffice Base?
4. Discuss the SQL view method of creating a query?
5. Describe the ways to refine the design of a report.

APPENDIX

Description of Commonly Used Functions

There are three types of functions that are used to set the Control Source property of calculated controls and/or to form part of calculated field expression in SQL statement. A brief description of the commonly used functions is below :

A1. Domain Aggregate Functions

These functions are used to perform calculations based on values in a field of a table or query. Criteria to select the set of records in the table or query that is desired to be used for calculations may also be specified. The criteria, if not specified, imply that all the records of the table or query specific to the field are used for computation. All the domain aggregate functions use the same syntax as is given

hereunder :

DFunction (“FldName”, “TblName” or QryName”, “SrchCond”)

Wherein DFunction refers to a named domain aggregate function. A brief description of its input arguments is given below:

FldName : It refers to the name of field that is to be searched in a table or query, which is specified as an argument.

TblName (or QueryName) : It refers to the name of a table or query that contains the field specified as second input argument.

SrchCond : It refers to the search condition on the basis of which the relevant record is searched.

Some of the important domain aggregate functions have been described as below :

(a) DLookup : This function is meant to look up information that is stored in a table or query, which is not the underlying source of Access Form or Report. It is used to set the Control Source property of a calculated control to display data from other table or query. Consider the following example:

DLookup (“Name”, “Accounts”, “Code = ‘110001’”)

In the above example, this function has been applied to search the name of account (in Accounts table) whose code is ‘110001’.

(b) DMax and DMin : These functions are used to retrieve respectively the

maximum and minimum values in the specified field. Consider the following example :

DMin (“Amount”, “Vouchers”, “Debit = ‘711001’”)

Dmax (“Amount”, “Vouchers”, “Debit = ‘711001’”)

In the above examples, the amount of minimum purchase transaction and maximum purchase transaction is retrieved and reported. It may also be noted that ‘711001’ is the code of Purchase account in Accounts table

(c) DSum : This function computes and returns the sum of the values in the specified field or expression. For Example, in a table : Sales that contains ItemCode, Price and Quantity as fields, the total amount of sales may be computed by using the DSum () function as follows :

DSum (“Price*Quantity”, “Sales”)

However, if the total sales is to computed for a particular item coded as 1678, the DSum () function shall be applied as follows :

DSum (“Price*Quantity”, “Sales”, “ItemCode = 1678”)

(d) DFirst and DLast : These functions are used to retrieve respectively the values in the specified field from first and last physical records. Consider the following application examples :

DFirst (“Name”, “Accounts”)

DLast (“Name”, “Accounts”)

In the above examples, the name first and last account that physically exists in Accounts table is retrieved and reported.

(e) DCount : This function is meant to compute the number of records with nonnull values in the specified field. Consider the following application example :

DCount (“*”, “Accounts”)

In the above example, The number of records in accounts table are counted and reported by DCount () function.

A2. SQL Aggregate Functions

The SQL aggregate functions have the functionality similar to that of domain aggregate function. However, unlike domain aggregate functions, these functions cannot be called directly into controls used in Forms and Reports of Access. These functions are used in SQL statements that provide the underlying record source of Forms and Reports. All these functions, when used require the GROUP BY clause in SQL statement :

(a) Sum : This function is used to compute and return the sum of a set of values. For Example, consider the following SQL statement that has been used in ChapterV to prepare the underlying information source of Trial Balance (ModelI.).


```
SELECT Debit As Code, Sum (Amount) As Total
FROM VOUCHERS
GROUP By Debit ;
```

In the above SQL statement, Sum () has been used to compute the total amount by which the transacted accounts have been debited.

(b) Min and Max : These functions are used to retrieve respectively the minimum and maximum of value set with respect to field or query expression. For Example, the following SQL statement is capable of returning the amount of minimum and maximum sales transaction in ModelI :

```
SELECT Min (Amount) As MinSales, Max (Amount) As MaxSales
FROM Vouchers
WHERE Credit = '811001' ;
```

It may be noted that the sales account that is coded as '811001' is credited as and when a sales transaction is recorded.

(c) Count : This function counts the number of records returned by a query. The number of times a sales transaction has occurred and recorded in books of accounts can be known by executing the following SQL statement.

SQL statement.

```
SELECT count (*)
FROM Vouchers
WHERE Credit = '811001'
```

In the above SQL statement, the Credit field stores the account code of sales when a sales transaction occurs. The WHERE clause restricts the number of records returned by the above SQL to those in which credit field has the account code of sales. Accordingly, the count () function returns the count value of records returned by the above SQL statement.

(d) First and Last : These functions are meant to retrieve the first and last record of a value set pertaining to a field or query expression.

A3. Other Functions

(a) IIF : The purpose of this function is to provide a value to the field from a mutually exclusive set of values. Its syntax is as given below :

```
IIF (<Condition>, Value1,
```

Value2)

Wherein <Condition> refers to any logical expression in which a comparison is made by using following comparison operators :

= equal to

<less than

>greater than

<= less than or equal to

>= greater than or equal to

The condition formed by the above comparison operators is evaluated to result into TRUE or FALSE.

<Value1>

This value is returned by IIF() function to the field, if the condition turns out to be TRUE

<Value2>

This value is returned by IIF() function to the field, if the condition turns out to be FALSE

Example : Suppose a field Type is to return the string of characters “Debit” when its value is 0 and “Credit” when its value is 1, IIF() function is used as shown below :

IIF (Type = 0, “Debit”, “Credit”)

(b) Abs : The purpose of this function is to return absolute value, This function receives a numeric value as its input argument and returns an absolute value. Consider the following examples on use of Abs () function :

When – 84 is given as input argument to Abs(– 84), it returns 84

When 84 is given as input argument to Abs(84), it returns 84

(c) Val : The purpose of this function is to return the numbers contained in a string as a numeric value of appropriate type. Its Syntax is Val(string). The string argument of the above Val() function is any valid string expression. The Val() function stops reading the string at the first character that cannot be recognised as number. For example, Val(“12431”) returns the value 12431 by converting the enclosed string of numerals into value. However, Val(“12,431”) returns the numeric value 12 because comma after 12 in the enclosed string of characters in Val () function is not recognised as number.