

**Practical Workbook of**

**CSC204-Database Systems**

Submitted by: Student Name (RollNo)

Submitted to: Ms.Zartasha Baloch

Department of Computer Systems Engineering

Mehran University of Engineering and Technology Jamshoro

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Of Contents With Rubrics** | | | | |
| **S.No** | **Lab Objectives** | **Taxonomy Level** | **Evaluation Criteria** | |
| **Quality/**  **Accuracy** | **Timeliness** |
| 1 | To create databases in Ms. Access and to create relationships between tables in a database. | P3 |  |  |
| 2 | Creating Forms and generating Reports on the database in MS Access. | P3 |  |  |
| 3 | To understand and practice Table creation and population of the table. | P3 |  |  |
| 4 | To retrieve data from the database using SQL SELECT Statement. | P3 |  |  |
| 5 | To use Single row functions in SQL queries. | P3 |  |  |
| 6 | To use Group functions in SQL queries. | P3 |  |  |
| 7 | To execute queries for retrieving data from multiple tables (using Joins). | P4 |  |  |
| 8 | To study and demonstrate SQL sub-queries. | P4 |  |  |
| 9 | To Update and Delete the contents of a Table and use of commit and rollback commands. | P4 |  |  |
| 10 | To demonstrate use of Indexes, Views & Sequences | P4 |  |  |
| 11 | To execute PL/SQL block using Control Structures and Data Types. | P4 |  |  |
| 12 | To explore and execute Exception Handling in PL/SQL | P4 |  |  |
| 13 | To create and use Cursors in PL/SQL | P4 |  |  |
| 14 | Working with Stored Functions and Procedures in PL/SQL | P4 |  |  |
| 15 | To create a database for a real-world scenario. | P4 |  |  |
|  | Total Marks of each Level | |  |  |

|  |  |
| --- | --- |
| Total Marks |  |
| Percentage |  |
| Marks Obtained /15 |  |

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 1** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To create databases in Ms. Access and to create relationships between tables in a database.

**Tools/Software:** Microsoft Access

**Introduction:**

A **database** is a collection of related information and has several objects such as a table, query, form or report.

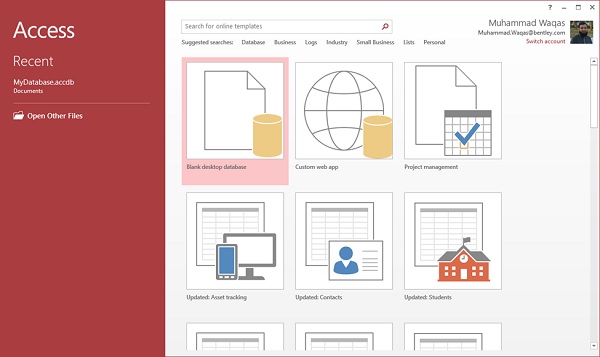
A **table** is a grouping of related data organized in fields (columns) and records (rows) on a datasheet. Many tables can be stored in a single database.

A **field** is a column on a datasheet and defines a data type for a set of values in a table. For a mailing list a table might include fields for first name, last name, address, city, state, zip code, and telephone number.

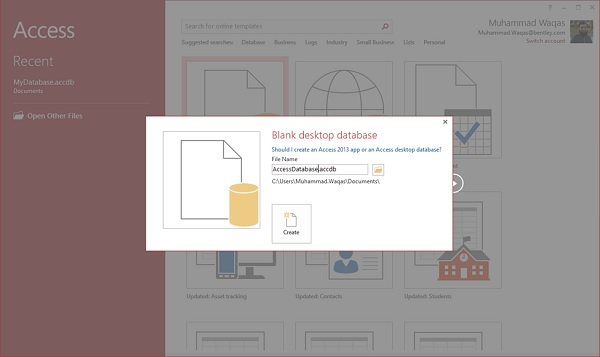
A **record** in a row on a datasheet and is a set of values defined by fields. In a mailing list table, each record would contain the data for one person as specified by the intersecting fields. Every record in a given table has the same fields in the same order.

**Steps to create a database in Ms. Access:**

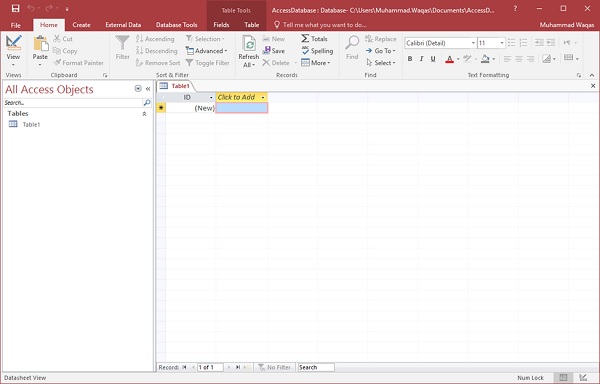
**Step 1** − Let us start by opening MS Access.



**Step 2** − Select Blank desktop database. Enter the name and click the Create button.



**Step 3** − Access will create a new blank database and will open up the table which is also completely blank.



Every field in a table has properties and these properties define the field's characteristics and behavior. The most important property for a field is its data type. A field's data type determines what kind of data it can store. MS Access supports different types of data, each with a specific purpose.

**Step 4**:

Let us now go to the Field tab and you will see that it is also automatically created.

The **ID** which is an AutoNumber field acts as our unique identifier and is the primary key for this table.

**Step 5:**

Click on the **Name & Caption** option in the Ribbon and Change the name of this field to make it more specific to this table. Enter the other optional information if you want and click Ok.

We can add some more fields by clicking on **click to add** and choose a suitable data type.

**Step 6:**

Once all the fields are added, click the Save icon. You will also see the **Save As** dialog box, where you can enter a table name for the table. We can create a primary key for this table by clicking on **Primary Key** option in the ribbon.

**Step 7**:

We have already created one table using **Datasheet View**. We can see its **Design View** by clicking on **CREATE** tab and tapping Design view.

**Exercise**

**Task 01:** Create tables to show following one to one (1:1) relationship. Insert at least 5 records in each table. A nurse may be in-charge of a care center. A care center must have one nurse in-charge.



**Task 02:** Create a database of University that consists of 03 tables (i.e. Student, Department, Teacher) & enter at least 05 records in each table. All three tables are related with each other.

**Task 03:** Create relationship among the three tables.

**Task 04:** Populate the tables with appropriate data.

**Task 05:** Write down a query for the Task #02 database, in which the user can find which teacher is teaching students of the Computer System Department.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 02** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** Creating Form and generating Reports on database in MS Access.**Tools:** MicroSoft Access

Forms in Access are like display cases in stores that make it easier to view or get the items that you want. Since forms are objects through which you or other users can add, edit, or display the data stored in your Access desktop database, the design of your form is an important aspect. There's a lot you can do design-wise with forms in Microsoft Access. You can create two basic types of forms −

* Bound forms
* Unbound forms

**Bound Forms**

Let us now understand what Bound Forms are −

* Bound forms are connected to some underlying data source such as a table, query, or SQL statement.
* Bound forms are what people typically think of when they think of the purpose of a form.
* Forms are to be filled out or used to enter or edit data in a database.
* Examples of bound forms will typically be what users use to enter, view or edit data in a database.

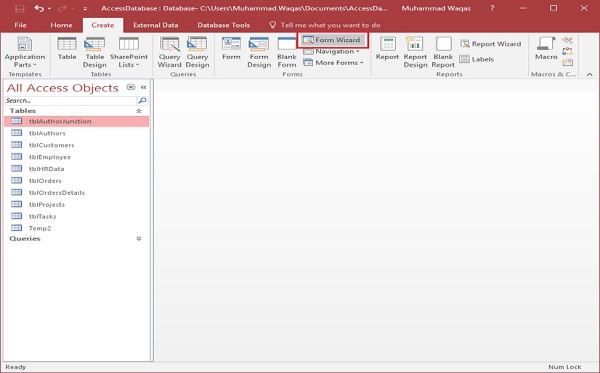
**Unbound Forms**

Let us look into Unbound Forms −

* These forms are not connected to an underlying record or data source.
* Unbound forms could be dialog boxes, switch boards, or navigation forms.
* In other words, unbound forms are typically used to navigate or interact with the database at large, as opposed to the data itself.

**Creating Forms**

There are a few methods you can use to create forms in Access. For this, open your Database and go to the **Create tab**. In the Forms group, in the upper right-hand corner you will see the Form Wizard button.

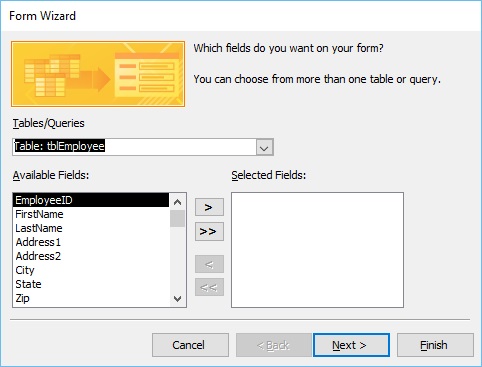


Click on that button to launch the Form Wizard.

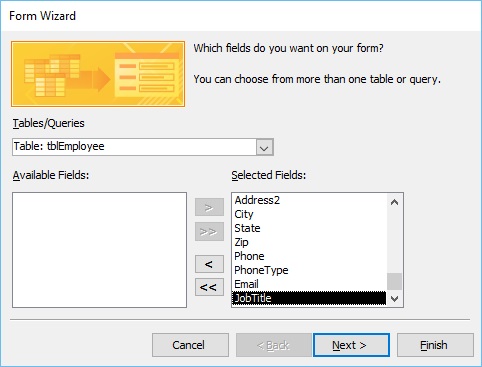
On this first screen in the wizard, you can select fields that you want to display on your form, and you can choose from fields from more than one table or a query.

Let us assume we want to simply have a quick form that we are going to use for data entry for our employee information.

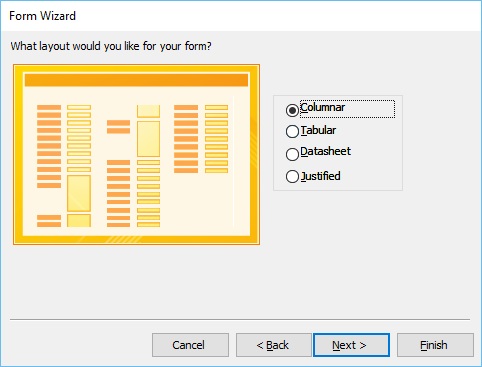
From **Tables/Queries** drop-down list, select **tblEmployees** table. Click on the double arrow to move all the fields at once.



Let us just leave it with that one table, and click **Next**.

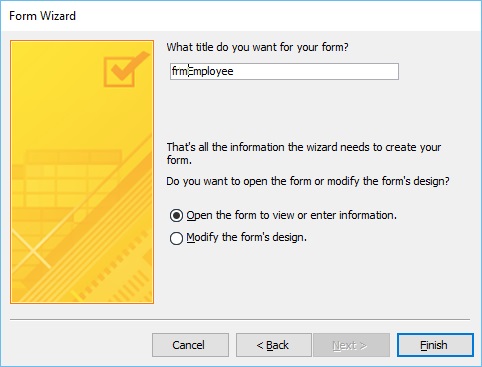


The following screen in the Form Wizard will ask for the layout that we would like for our form. We have **columnar, tabular, datasheet and justified** layouts. We will choose the columnar layout here and then click **Next**.

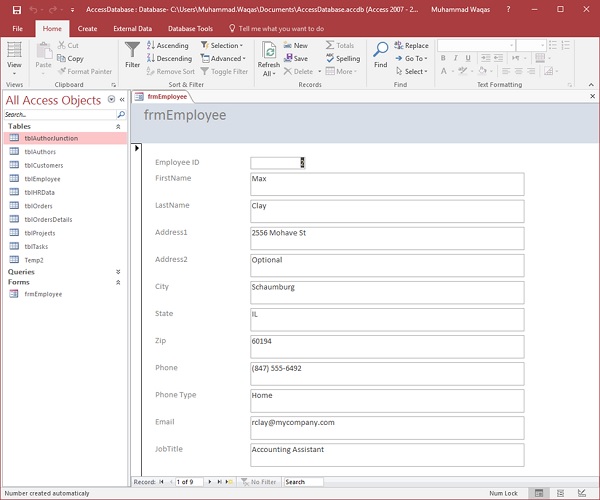


In the following screen, we need to give a title for our form. Let us call it **frmEmployees**.

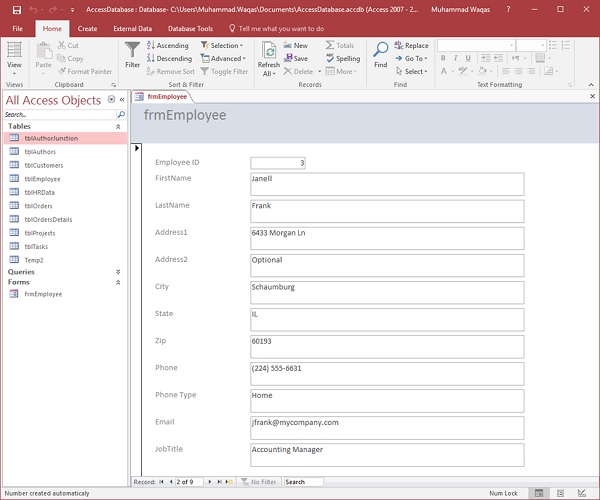
Once you have given your form a title, you can open the form to see what that form looks like, or you can begin entering information into your table. Or you can choose the option to modify the form's design. Let us choose the first option to **open the form to view or enter information** and click **Finish**.

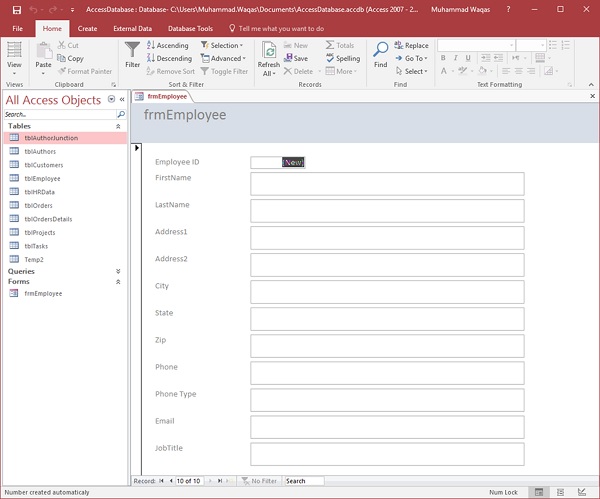


Now, take a look at the following screenshot. This is what your form looks like. This is a single item form, meaning one record is displayed at a time and further down you can see the navigation buttons, which is telling us that this is displaying the record 1 of 9. If you click on that button then, it will move to the next record.



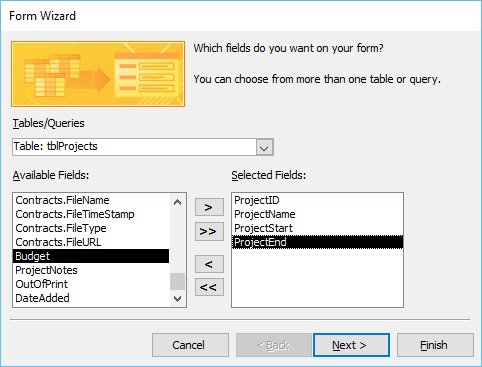
If you want to jump to the very last record in that form or that table, you can use the button right beside that right arrow, the arrow with a line after it, that's the last record button. If you want to add new employee information, go to the end of this records and then after 9 records you will see a blank form where you can begin entering out the new employee's information.



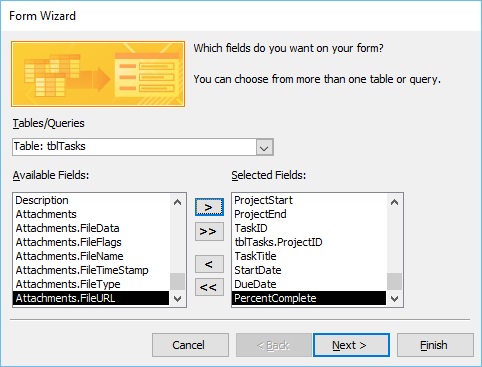


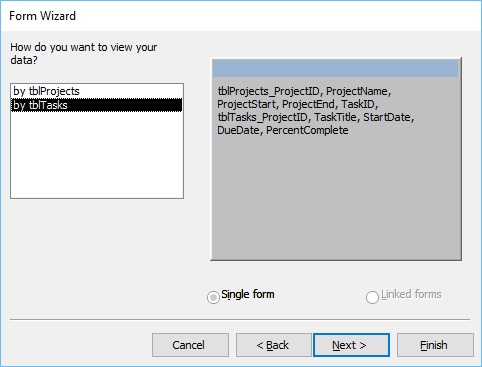
This is one example of how you can create a form using the Form Wizard. Let us now close this form and go to the Create tab. Now we will create a slightly more complicated form using Wizard. Click the Form Wizard and this time, we will choose fields from a couple of different tables.

In this Form Wizard, let us choose **tblProjects** for **Tables/Queries**, and select a few Available Fields such as ProjectID, ProjectName, ProjectStart, and ProjectEnd. These fields will now move to Selected Fields.



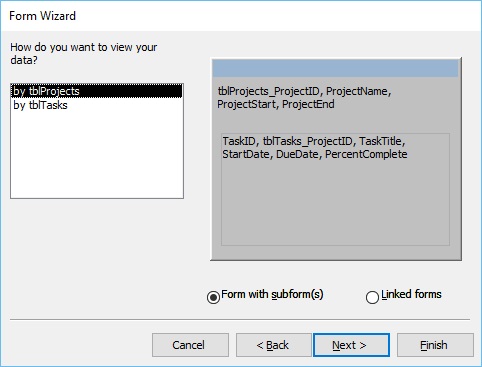
Now select **tblTasks** for Tables/Queries and send over the TaskID, ProjectID, TaskTitle, StartDate, DueDate and PercentComplete. Click **Next**.





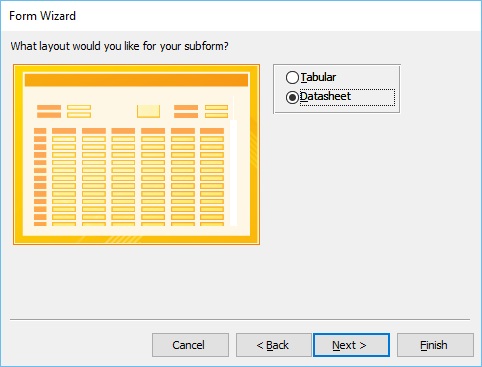
Here, we want to retrieve data from a couple of different objects. We can also choose from options on how we want to arrange our form. If we want to create a flat form, we can choose to arrange by **tblTasks**, which will create that single form, with all the fields laid out in flat view as shown above.

However, if we want to create a hierarchical form based on that one-to-many relationship, we can choose to arrange our data by tblProjects.

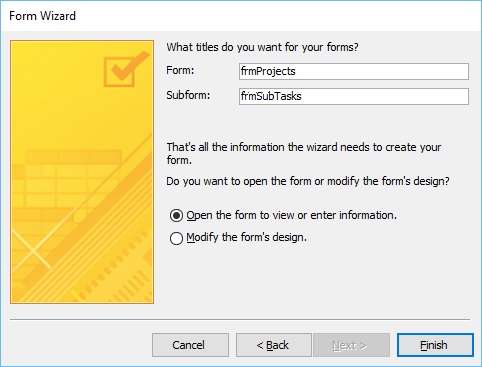


In the above window, we have the option to include a **subform** for **tblTasks**, or we can make that a linked form. This linked form is where tblProjects will have a button that will launch that second form filtered to the project that we have selected in that underlying projects form. Let us now select the **Form with subform(s)**, and then click **Next**.

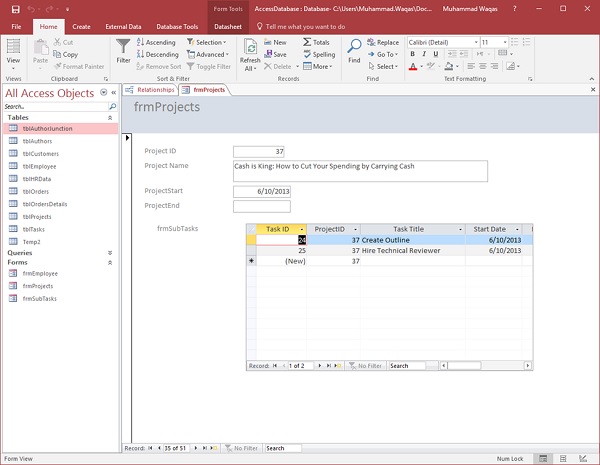
In the following screen, you can choose a layout for your subform. The Datasheet View gets selected by default. The Datasheet View is similar to Table View. Now, click **Next**.



In the following screen, you need to provide a name for your forms. Enter the name you want and click **Finish**.

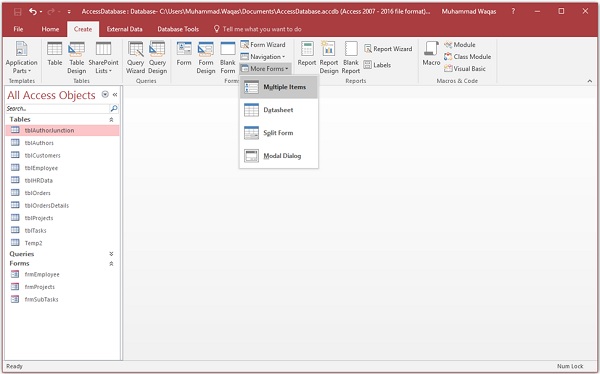


Access will give you a preview of what your form looks like. On top, you have the controls on your main form, which is from our **Projects** table. As you go down, you will see a subform. It's like a form within a form.



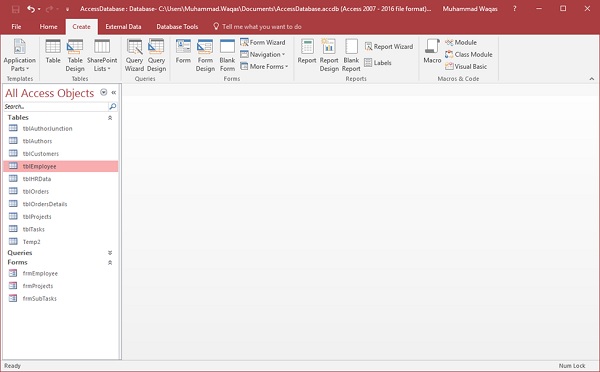
Multiple Item Form

You may also want to create a specific kind of form. For this, you can click on the **More Forms** drop-down menu.

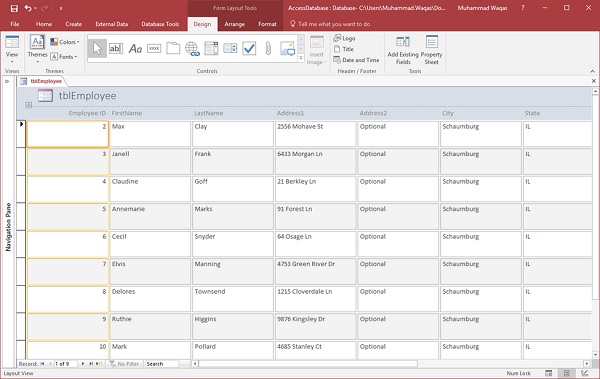


From the menu, you can create a **Multiple Items** form, a **Datasheet** form, a **Split** form, or even a **Modal Dialog** form. These are typically bound forms; select the object that you would like to be bound to that form. This does not apply to the Modal Dialog forms.

To create this type of form, you will need to select the object in navigation pane first. Let us select **tblEmployees** here.



Proceed by clicking on **More Forms** and **Multiple Items**.

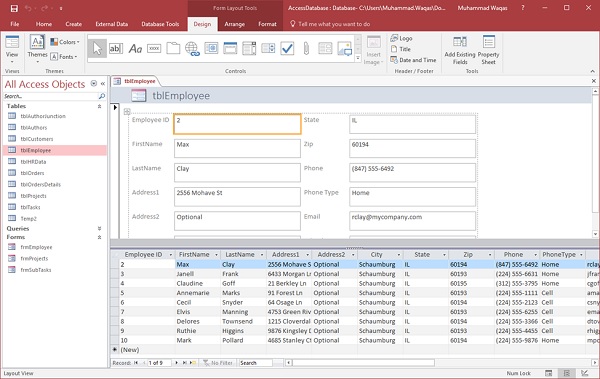


The above step will further create a Multiple Items form, listing out all the employees.

Split Form

This type of form is divided in equal halves, either vertically or horizontally. One half displays a single item or record, and the other half displays a list or a datasheet view of multiple records from the underlying data source.

Let us now select **tblEmployees** in the navigation pane and then on **Create** tab. Select **Split Form** option from More Forms menu and you will see the following form in which the form is divided vertically.



**Exercise**

1. Create forms for each table created in lab#1.
2. Generate reports for each table created in lab#1.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 03** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To be familiar with Table creation and population of data in a table in SQL.

**Tools:** MySQL workbench/ORACLE SQL

**Table Creation:-**

For Creating a table CREATE TABLE Command is used. This statement is a part of Data Definition Language (DDL). The typical syntax for CREATE TABLE command is:

**syntax:**

*CREATE TABLE*table\_name *(*    column1 datatype*,*    column2 datatype*,*    column3 datatype*,  
   ....  
);*

**Example:-**

Create table member (

MemberId varchar2 (7),

Password varchar2 (8) not null,

Fname vrachar2 (15) not null,

Lname vrachar2 (15) not null,

Address vrachar2 (50),

Email vrachar2 (15),

Cash balance number(10,2) not null,

Primary key (MemberId)

);

**Data Population:-**

This section introduces the SQL INSERT statement that adds a row to the table in the database.

**Syntax:**

*INSERT INTO <table name>[(column {, column})]*

*VALUES (expression{, expression});*

**Example:**

Insert into member values (

96, ‘tiger’,’saloon’,’hilux’,”B #32,stechin road, england”,” ”, 76000);

**Create Table Using Another Table**

A copy of an existing table can also be created using CREATE TABLE. The new table gets the same column definitions. All columns or specific columns can be selected. If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

**Syntax**

CREATE TABLE *new\_table\_name* AS  
    SELECT *column1, column2,...*  
    FROM *existing\_table\_name*  
    WHERE ....;

**Renaming a Table**

There are two ways to rename tables. The first one uses the ALTER TABLE syntax:

ALTER TABLE old\_table\_name RENAME new\_table\_name;

The second way is to use RENAME TABLE:

RENAME TABLE old\_table\_name TO new\_table\_name;

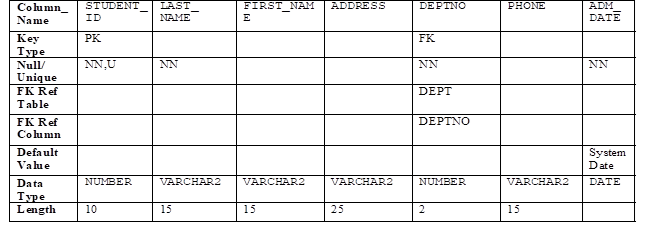
**RENAME TABLE** offers more flexibility. It allows renaming multiple tables in one statement.

**Exercise**

**Task 01:** Create department table with field names department id department name and location, take data types accordingly.

**Task 02:** Create Employee table with columns Employee No, employee name, job, mgr, hire date, salary, commission, department id. Make employee no as primary key and department id as foreign key.

**Task 03:** Create Student table based on the following table instance chart. Choose the appropriate data types and be sure to add integrity constraints.

****

**Task 04:** Add at least 5 values of your choice in each table.

**Task 05:** Create employee2 table based on the structure of employee table.

**Task 06:** Create table My\_Employee based on structure of employee table. Include columns employee no, employee name and salary. Name them in new table as id, Name and Salary.

**Task 07:** Rename employee2 table employee.

**Task 08:** Drop employee table.

**Task 09:** Add comments to student table describing the table.

**Task 10:** Add comment on admission date column as ‘Admission date of students’

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 04** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To retrieve data from tables of a database using SQL SELECT Statement.

**Tools**: ORACLE/MySQL

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

**Syntax:**

SELECT [DISTINCT] *{\*|*column1, column2, ...}  
FROM table\_name  
WHERE condition

ORDER BY column1, column2, ... ASC|DESC;

The DISTINCT statement is used to return only distinct (different) values.

The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified condition.

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

**Example:**

SELECT last\_name, job\_id, salary

FROM employees

WHERE (job\_id = 'SA\_REP'

OR job\_id = 'AD\_PRES')

AND salary > 15000

ORDER BY last\_name;

**Exercise**

**Consider following tables to write SQL queries.**

**EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO )**

**DEPT (DEPTNO, DNAME, LOC)**

1. Write a query to display unique departments from EMP table.
2. Display the employee name, job, and joining date of employees hired between February 20, 1981 and May 1, 1981. Order the query in ascending order by joining date.
3. Display the names of all employees who have two ‘**L**’ in their name and are in department 30 or their manager is 7782.
4. Display the name, salary, and commission for all employees whose commission amount is greater than their salary increased by 10%.
5. Display the name and salary of employees who earn more than 1500 and are in department 10 or 30. Label the columns Employee and Monthly Salary, respectively.
6. List out the employees whose name start with **S** and ends with **H**.
7. Write a query that produces following for each employee:

<employee name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

1. Display the name job and salary for all employees whose job is clerk or analyst and their salary is not equal to 1000, 3000, or 5000.
2. Display the names of all employees where the third letter of their name is an A.
3. Display name, salary and commission for all employees who earn commission. Sort the result in descending order of salary and commission.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 05** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To use Single row functions in SQL queries.

**Tools:** ORACLE LIVE SQL

**Single Row functions:**

Single row functions work on a single row and return one output per row. For example, length and case conversion functions are single row functions.

**Types:**

* **General functions**: Usually contains NULL handling functions. The functions under the category are NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.
* **Case Conversion functions**: Accepts character input and returns a character value. Functions under the category are UPPER, LOWER, and INITCAP.
* **Character functions**: Accepts character input and returns number or character value. Functions under the category are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM, and REPLACE.
* **Date functions:** Date arithmetic operations return date or numeric values. Functions under the category are MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY, LAST\_DAY, ROUND, and TRUNC.
* **Number functions:** Accepts numeric input and returns numeric values. Functions under the category are ROUND, TRUNC, and MOD.

**Example:**

SELECT UPPER(last\_name), TO\_CHAR(hire\_date, 'fmDD Month YYYY') AS HIREDATE

FROM employees;

**Exercise**

1. Display the employee name and their annual salary including commission amount. If the commission is null replace it with 0.
2. Display Employee’s name with the first letter capitalized and all other letters lowercase and the length of their name, for all employees whose name starts with J, A, or M. Give each column appropriate names.
3. Display the name, hiredate, and day of the week on which the employee started. Label the column **First Day**.
4. For each employee, display the employee name and calculate the number of months between today and the date the employee was hired. Label the column MONTHS\_WORKED. Order the result by the number of months employed. Round the number of months up to the closest whole number.
5. Display the Employee number, name, salary, and salary increase by 15% expressed as a whole number. Label the column New Salary.
6. Display the employee name, hiredate, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to “Sunday, the seventh of September, 1981”.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 06** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To use Group functions in SQL queries.

**Tools:** Oracle Live SQL

**GROUP BY functions:**

Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group.

**Types:**

* **COUNT ():** This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition. If the WHERE condition is not specified, then the query returns the total number of rows in the table.
* **MAX():** This function is used to get the maximum value from a column.
* **MIN():** This function is used to get the minimum value from a column.
* **AVG():** This function is used to get the average value of a numeric column.
* **SUM():** This function is used to get the sum of a numeric column

**Example:**

SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary)

FROM employees

WHERE job\_id LIKE '%REP%';

**Exercise**

1. List out the department numbers that have at least 4 employees.
2. Display the number of employees in each department.
3. Display the manager no and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is less than 1000. Sort the output in descending order of salary.
4. Find the most recently hired employee in each department.
5. List the highest salary paid for each job.
6. Display the department number, number of employees in that dept and the average salary for all employees in that department. Round the average salary to two decimal places.
7. Write a query that will display the difference between the highest and lowest salaries. Label the column DIFFERENCE.
8. Display the number of employees with same job.
9. Determine the number of managers without listing them. (Hint: Use MGR column)
10. Display the job title and total monthly salary for each job title with a total payroll exceeding 5000. Exclude salespeople and sorts the list by the total monthly salary.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 07** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To retrieve data from multiple tables (using Joins).

**Tools:** Oracle/MySQL

**JOINS:**

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is equal to symbol.

There are different types of joins available in SQL −

* INNER JOIN − returns rows when there is a match in both tables.
* LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
* RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
* FULL JOIN − returns rows when there is a match in one of the tables.
* SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
* CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

**Example:**

SELECT e.employee\_id, e.last\_name, e.department\_id,

d.department\_id, d.location\_id

FROM employees e , departments d

WHERE e.department\_id = d.department\_id;

**Exercise**

1. List the name of the employees with the name of their immediate higher authority.
2. Create a unique listing of all jobs that are in department 30. Include the location of department 30 in the output.
3. Display the employee name and department name for all employees who have an A in their name.
4. Display the employee name and department name for all employees who work in DALLAS.
5. Display the employee name employee number along with their manager name and manager’s number for all employees including KING, who has no manager. Label the columns employee, Emp#, Manager, and Mgr#, respectively.
6. Create a query that will display the name, job, department name, salary, grade for all employees.
7. Create a query to display the name and hire date of any employee hired after employee BLAKE.
8. Display all employees’ names and hire dates along with their manager’s name and hire date for all employees who were hired before their managers. Label the columns Employee, Emp Hiredate, Manager, and Mgr Hiredate, respectively.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 08** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To study and practice SQL sub-queries.

**Tools:** Oracle/MySQL

**SUB-QUERY:**

A Subquery or Inner query or a Nested query in SQL is a query inside another SQL query and inserted inside the WHERE clause.

A SQL Subquery is used to return information that will be used in the primary query as a condition to additionally limit the information to be recovered.

**SYNTAX:**

SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

WHERE column\_name OPERATOR

(SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

[WHERE] );

**Example:**

SELECT last\_name

FROM employees

WHERE salary >

(SELECT salary

FROM employees

WHERE last\_name = 'Abel');

**Exercise**

1. Display the employee name and hire date for all employees in the same department as BLAKE. Exclude blake.
2. Display the employee name and salary for all employees who earn more than average salary.
3. Display the employee name, job and hire date for all employees who report to KING.
4. List the employee details whose salary is greater than the lowest salary of an employee belonging to deptno 20.
5. Which department has the highest Monthly remuneration bill (Salaries of employees)?
6. Display the employees that earn a salary that is higher than the salary of all the clerks. Sort the result on salary from highest to lowest.
7. Create a query to display the name, hire date and salary for all employees who have both the same salary and commission as employee SCOTT.
8. Display the names and salaries of those employees who earn highest salary in their department.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 09** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To Update and Delete the contents of a Table and use commit and rollback commands for transaction control.

**Tools:** Oracle/MySQL

**INSERT STATEMENT:**

The INSERT statement inserts rows into an existing table.

The simplest recommended form of the INSERT statement has this syntax:

INSERT INTO table\_name (list\_of\_columns)

VALUES (list\_of\_values);

Every column in list\_of\_columns must have a valid value in the corresponding position in list\_of\_values. Therefore, before you insert a row into a table, you must know what columns the table has, and what their valid values are.

**UPDATE STATEMENT:**

The UPDATE statement updates (changes the values of) a set of existing table rows.

A simple form of the UPDATE statement has this syntax:

UPDATE table\_name

SET column\_name = value [, column\_name = value]...

[ WHERE condition ];

Each value must be valid for its column\_name. If you include the WHERE clause, the statement updates column values only in rows that satisfy condition.

**DELETE STATEMENT:**

The DELETE statement deletes rows from a table.

A simple form of the DELETE statement has this syntax:

DELETE FROM table\_name [ WHERE condition ];

If you include the WHERE clause, the statement deletes only rows that satisfy condition. If you omit the WHERE clause, the statement deletes all rows from the table, but the empty table still exists. To delete a table, use the DROP TABLE statement.

**EXERCISE:**

1. Create My\_Employee table whose structure and data is same as EMP table. Use My\_Employee table for following queries.
2. Make the data additions permanent.
3. Change the name of employee 7722 to Drexler.
4. Change the salary to 1000 for all employees with a salary less than 900.
5. Delete employee Allen from the MY\_EMPLOYEE table.
6. Mark an intermediate point as A in the processing of the transaction.
7. Empty the entire table.
8. Discard the most recent DELETE statement without discarding the earlier statements.
9. Save the changing permanently.
10. Update the salary of employee 100 to 5000.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 10** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To demonstrate use of Indexes, Views & Sequences

**Tools:** Oracle Live SQL

**INDEXES:**

An index is a schema object that has the role to provide direct and fast access without reading the entire table. Indexes are created explicitly or automatically.

**VIEWS:**

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is a composition of a table in the form of a predefined SQL query. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

**SEQUENCES:**

A sequence is a user-defined schema-bound object that generates a sequence of numeric values. Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

Syntax:

|  |  |
| --- | --- |
| INDEX | CREATE INDEX index\_name  ON table\_name(column\_name); |
| VIEW | CREATE VIEW view\_name AS  SELECT column1, column2......  FROM table\_name  WHERE [condition]; |
| SEQUENCE | CREATE SEQUENCE sequence\_name  START WITH initial\_value  INCREMENT BY increment\_value  MINVALUE minimum value  MAXVALUE maximum value  CYCLE|NOCYCLE; |

**Exercise**

1. Create a view called emp\_vu based on the employee number, employee name, and department number from the EMP table. Change the heading for the employee name to EMPLOYEE.
2. Display the contents of the EMP\_VU view.
3. Select the view name and text from the data dictionary USER\_views.
4. Create a view named DEPT\_VU\_20 that contains the employee number, employee name, and department number for all employees in department 20. Label the view column Employee\_Id, Employee, and Department\_Id. Do not allow an employee to be reassigned to another department through the view.
5. Attempt to reassign Smith to department 30.
6. Create a view Salary\_vu based on the employee name, department name, salary, and salary grade for all employees. Label the columns Employee, Department, Salary, and Grade.
7. Create a sequence to be used with the primary key column of the department table. The sequence should start at 60 and have a maximum value of 200. Have your sequence increment by ten numbers. Name the sequence Dept\_Id\_Seq.
8. Display the following information about your sequences: sequence name, max value, increment size, and last number.
9. Create a non-unique index on the foreign key column in the employee table.
10. Create a synonym for Dept\_Id\_Seq.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 11** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To understand and practice PL/SQL block structure, Control Structures and

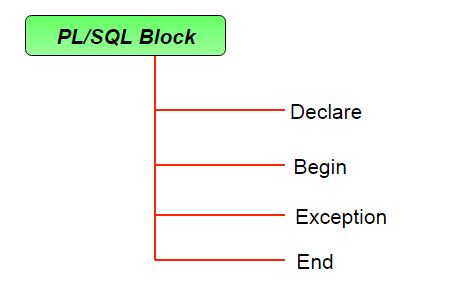
Data Types.

**Tools:** Oracle Live SQL

**PL/SQL AND ITS BLOCK STRUCTURE:**

PL/SQL is a block-structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to the oracle engine all at once which increases processing speed and decreases the traffic.

The basic unit in PL/SQL is a block. All PL/SQL programs are made up of blocks, which can be nested within each other.



**Example:**

DECLARE

v\_deptno NUMBER(2);

v\_location VARCHAR2(13).

BEGIN

SELECT deptno, loc

INTO v\_deptno, v\_location

FROM dept

WHERE dname = 'SALES';

DBMS\_OUTPUT.PUT\_LINE(“The location for Department ”|| v\_deptno || “ is ”|| v\_location);

END;

**Exercise**

1. Write a PL/SQL block to calculate the annual salary of an employee whose ID is 7722.
2. Write a PL/SQL block to show the operator precedence and parentheses in 5 or more complex expressions.
3. Write a PL/SQL program to arrange the number of two variables in such a way that the small number will store in the num\_small variable and the large number will store in the num\_large variable.
4. Write a PL/SQL program to count the number of employees in department 50 and check whether this department has any vacancies or not. There is a total of 45 posts in this department.
5. Write a program in PL/SQL to check whether a number is prime or not using for loop.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 12** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To Explore Exception Handling In Pl/Sql

**Tools:** Oracle Live SQL

**EXCEPTION:**

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition.

There are two types of exceptions −

1. System-defined exceptions
2. User-defined exceptions

**Syntax for Exception Handling**

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements

WHEN exception2 THEN

exception2-handling-statements

........

WHEN others THEN

exception-handling-statements

END;

The default exception will be handled using WHEN others THEN.

**Example:**

DECLARE

c\_id customers.id%type := 8;

c\_name customerS.Name%type;

c\_addr customers.address%type;

BEGIN

SELECT name, address INTO c\_name, c\_addr

FROM customers

WHERE id = c\_id;

DBMS\_OUTPUT.PUT\_LINE ('Name: '|| c\_name);

DBMS\_OUTPUT.PUT\_LINE ('Address: ' || c\_addr);

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such customer!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

/

**Exercise**

1. Write a PL/SQL block that updates description of a product and raises a user-defined exception when that product is not found.
2. Write a simple PL/SQL code block, to demonstrate the use of Named Exception Handler for division by zero error.
3. Write a PL/SQL block for displaying information of a customer ID, when the user enters an invalid ID, raise the exception invalid\_id.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 13** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To create and use Cursors in PL/SQL.

**Tools:** Oracle Live SQL

**CURSORS:**

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

* Implicit cursors
* Explicit cursors

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as

1. %FOUND
2. %ISOPEN,
3. %NOTFOUND
4. %ROWCOUNT

The SQL cursor has additional attributes, %BULK\_ROWCOUNT and %BULK\_EXCEPTIONS, designed for use with the FORALL statement.

**SYNTAX:**

The syntax for creating an explicit cursor is −

CURSOR cursor\_name IS select\_statement;

**Example:**

DECLARE

c\_id customers.id%type;

c\_name customer.name%type;

c\_addr customers.address%type;

CURSOR c\_customers is

SELECT id, name, address FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers into c\_id, c\_name, c\_addr;

EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr);

END LOOP;

CLOSE c\_customers;

END;

/

**Exercise**

1. Write a program in PL/SQL to find the number of rows effected using SQL%ROWCOUNT attributes of an implicit cursor.
2. Write a program in PL/SQL to display detail information for the employee of ID 149 from the employees table.
3. Write a program in PL/SQL to display detail information of all employees from employees table using explicit cursor.
4. Write a PL/SQL block that uses explicit cursors to retrieve employees one by one and displays the name and salary of those employees currently working in deptno 30.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 14** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** Working with Stored Functions and Procedures in PL/SQL.

**Tools:** Oracle Live SQL

**STORED FUNCTIONS AND PROCEDURES:**

A subprogram created inside a package is a **packaged subprogram**. It is stored in the database and can be deleted only when the package is deleted with the DROP PACKAGE statement.

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms −

* **Functions** − These subprograms return a single value; mainly used to compute and return a value.
* **Procedures** − These subprograms do not return a value directly; mainly used to perform an action.

**SYNTAX:**

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

< procedure\_body >

END procedure\_name;

|  |
| --- |
| CREATE [OR REPLACE] FUNCTION function\_name  [(parameter\_name [IN | OUT | IN OUT] type [, ...])]  RETURN return\_datatype  {IS | AS}  BEGIN  < function\_body >  END [function\_name]; |

**Example:**

CREATE OR REPLACE FUNCTION get\_sal

(id employees.employee\_id%TYPE) RETURN NUMBER IS

sal employees.salary%TYPE := 0;

BEGIN

SELECT salary

INTO sal

FROM employees

WHERE employee\_id = id;

RETURN sal;

END get\_sal;

/

**Exercis** **e**

1. Create a stored function to calculate the salary ranking of the employee based on the current minimum and maximum salaries for employees in the same job category. (Hint: salary ranking = (empsal-minsal)/(Maxsal-Minsal)).
2. Write a stored procedure, that accepts radius of circle and displays area of the circle.
3. Write a stored function that accepts a number and returns it’s factorial.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Systems (3rd Semester)**  **Lab Experiment 15** |

|  |  |
| --- | --- |
| **Roll No:** |  |
| **Date of Conduct:** |  |
| **Submission Date:** |  |

|  |  |  |
| --- | --- | --- |
| **Rubrics** | **Marks Obtained** | |
| 0 | 1 |
| **Completeness & Accuracy** |  |  |
| **Timelines** |  |  |

**Objective:** To create a database for a real-world problem.

**Tools:**