|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 1** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MicroSoft Access**

**Introduction:**

**MS ACCESS:** Microsoft Access is a Database Management System (DBMS) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools

* Microsoft Access is just one part of Microsoft’s overall data management product strategy.
* It stores data in its own format based on the Access Jet Database Engine.
* Like relational databases, Microsoft Access also allows you to link related information easily. For example, customer and order data. However, Access 2013 also complements other database products because it has several powerful connectivity features.

Microsoft Access stores information which is called a database. To use MS Access, you will need to follow these four steps −

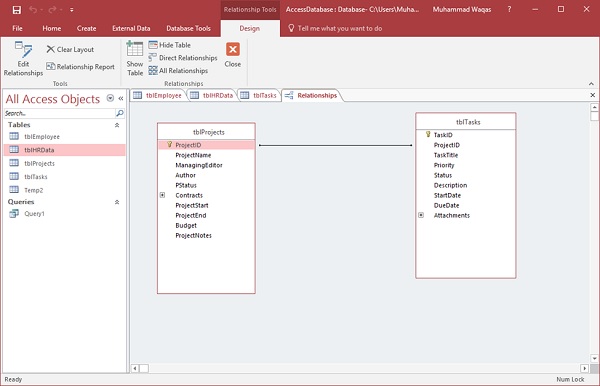
* **Database Creation** − Create your Microsoft Access database and specify what kind of data you will be storing.
* **Data Input** −After your database is created, the data of every business day can be entered into the Access database.
* **Query** −This is a fancy term to basically describe the process of retrieving information from the database.
* **Report** (optional) − Information from the database is organized in a nice presentation that can be printed in an Access Report.

## Why Create Table Relationships?

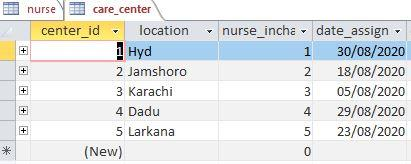
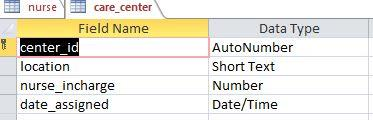
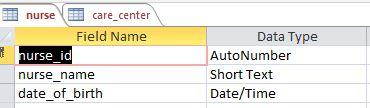
MS Access uses table relationships to join tables when you need to use them in a database object. There are several reasons why you should create table relationships before you create other database objects, such as forms, queries, macros, and reports.

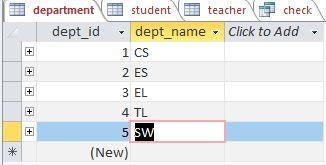
* To work with records from more than one table, you often must create a query that joins the tables.
* The query works by matching the values in the primary key field of the first table with a foreign key field in the second table.
* When you design a database, you divide your information into tables, each of which has a primary key and then add foreign keys to related tables that reference those primary keys.
* These foreign **key-primary key pairings** form the basis for table relationships and multi-table queries.

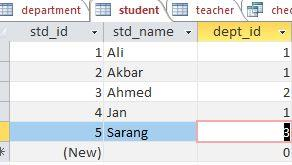
**Example Relationships Database.**



**Lab Task**

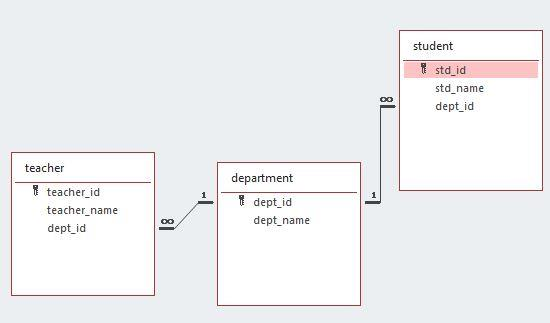
**********1. Create tables to show following one to one (1:1) relationship. Insert at least 5 records in each table. A nurse may be in-charge of a care center. A care center must have one nurse in-charge.**

2. **Create a database of University that consists of 03 tables (i.e. Student, Department, Teacher) and select appropriate primary key in each table.**





**3. Create relationship among the three tables.**



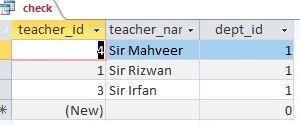
**4. Populate the tables with appropriate data.**

****

**5. Write down a query of TASK#02 database, in which user can find that which teacher is teaching students of Computer System Department.**

**Query:**



**Result of Query:**

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 2** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools:** MicroSoft Access

**Introduction:**

**Create a form from an existing table or query in Access:**

To create a form from a table or query in your database, in the Navigation Pane, click the table or query that contains the data for your form, and on the Create tab, click Form.

Access creates a form and displays it in Layout view. You can make design changes like adjusting the size of the text boxes to fit the data, if necessary. For more information, see the article on using the form tool.

**Create a blank form in Access**

1. To create a form with no controls or preformatted elements: On the Create tab, click Blank Form. Access opens a blank form in Layout view, and displays the Field List pane.
2. In the Field List pane, click the plus sign (+) next to the table or tables that contain the fields that you want to see on the form.
3. To add a field to the form, double-click it or drag it onto the form. To add several fields at once, hold down CTRL and click several fields, and then drag them onto the form at the same time.

**Note:** The order of the tables in the Field List pane can change, depending on which part of the form is currently selected. If you are not able to add a field to the form, try selecting a different part of the form and then try adding the field again.

1. Use the tools in the Controls group on the Form Layout Tools tab to add a logo, title, page numbers, or the date and time to the form.
2. If you want to add a wider variety of controls to the form, click Design and use the tools in the Controls group.

## Creating Reports:

Reports organize and summarize data for viewing online or for printing. A detail report displays all of the selected records. You can include summary data such as totals, counts, and percentages in a detail report. A summary report does not list the selected records but instead summarizes the data and presents totals, counts, percentages, or other summary data only. Access has several report generation tools that you can use to create both detail and summary reports quickly.

**To use the Report button:**

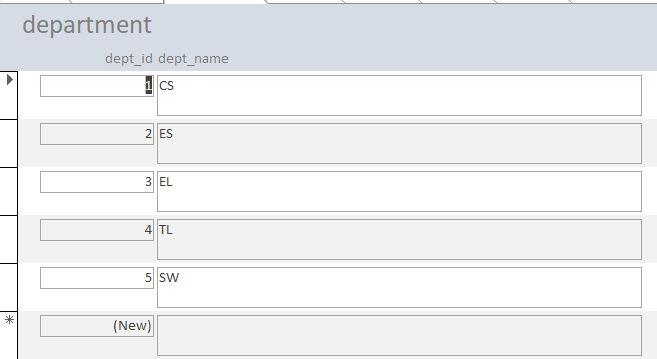
1. Open the Navigation pane.
2. Click the table or query on which you want to base your report.
3. Activate the Create tab.
4. Click the Report button in the Reports group. Access creates your report and displays your report in Layout view. You can modify the report.

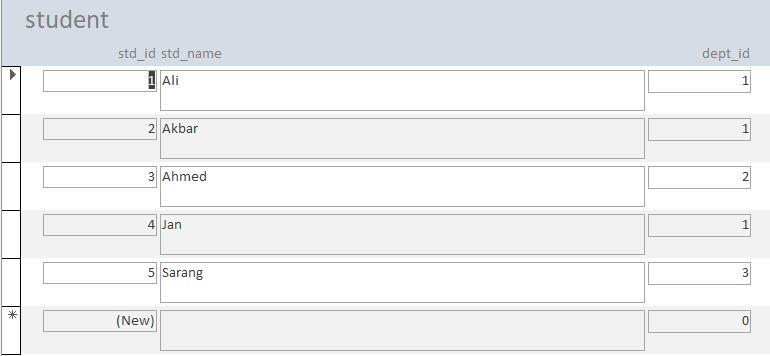
#### **To create a report by using the Report Wizard:**

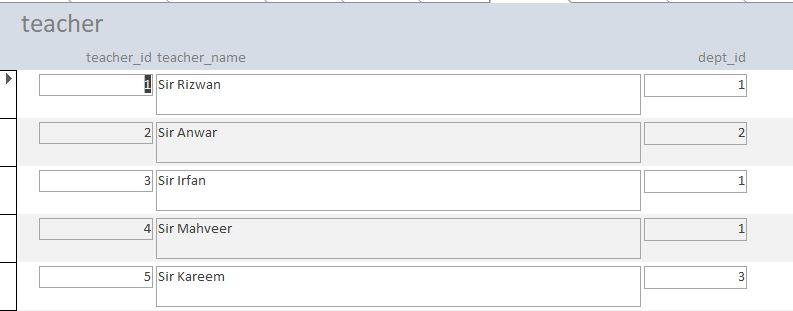
##### **Open the Report Wizard**

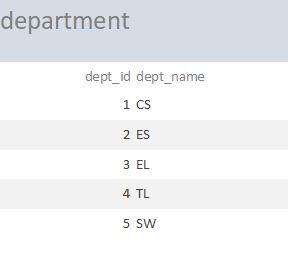
1. Activate the Create tab.
2. Click Report Wizard in the Reports group. The Report Wizard appears.

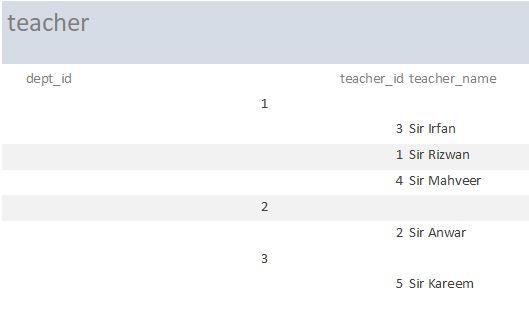
**Lab Task**

1. **Create forms for each table created in lab#1.**





1. **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 Management Systems (4th Semester) 18CS**  **Lab Experiment 3** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

**Objective: To be familiar with Table creation and population of table.**

**Tools: MySql / Oracle.**

**Creating Databases and Tables**

In order to be able to add and manipulate data, you first have to create a database. There’s not much to this. You’re creating just a container in which you will add tables. Creating a table is more involved and offers many choices.

There are a few basic things to decide when creating a structure for your data:

* The number of tables to include in your database, as well as the table names
* For each table, the number of columns it should contain, as well as the column names
* For each column, what kind of data is to be stored.

## Creating a Database:

## Creating a database is simple, mostly because there’s nothing much to it. Use the SQL statement CREATE DATABASE. You will have to provide a name for the database with this SQL statement. You could call it something bland like db1.

CREATE DATABASE rookery;

## Creating Tables:

The next step for structuring a database is to create tables. Although this can be complicated, we’ll keep it simple to start. We’ll initially create one main table and two smaller tables for reference information. The main table will have a bunch of columns, but the reference tables will have only a few columns.

**CREATE** **TABLE** birds (

Bird\_id **INT** AUTO\_INCREMENT **PRIMARY** **KEY**,

Scientific\_name **VARCHAR** (255) **UNIQUE**,

Common\_name **VARCHAR** (50),

Family\_id **INT**,

Description **TEXT**);

This SQL statement creates the table birds with five fields, or columns, with commas separating the information about each column.

The names of the columns can be anything other than words that are reserved for SQL statements, clauses, and functions. Actually, you can use a reserve word, but it must always be given within quotes to distinguish it.

## Adding data to your table

Let's inject some data into that table. We'll add information for a fictional member of the editorial staff. The data to be added is:

* Name: Olivia
* ID: 01
* Email: olivia@company.com

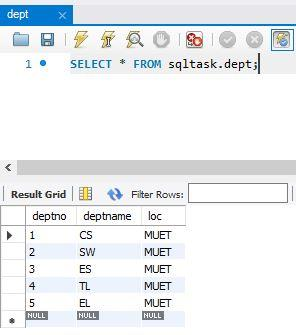
The command to add this would be:

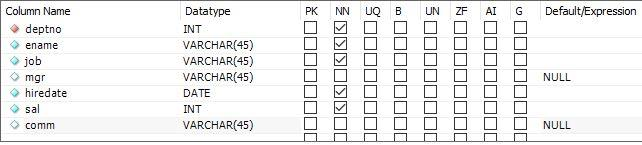
INSERT INTO editorial (id, name, email) VALUES (01,"Olivia","olivia@company.com");

You can view the information added to the table with the command:

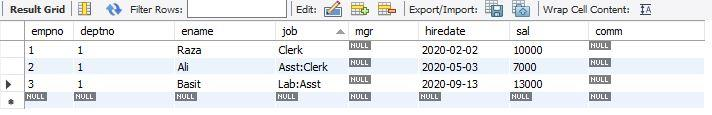
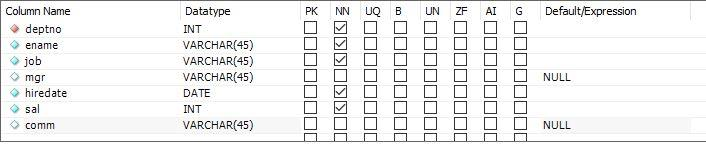
**Lab Task**

1. **Create the DEPT table with field names DEPTNO, DNAME and LOCATION. Take datatypes accordingly.**

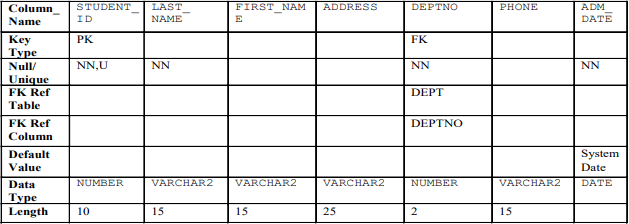
 **Solution:**

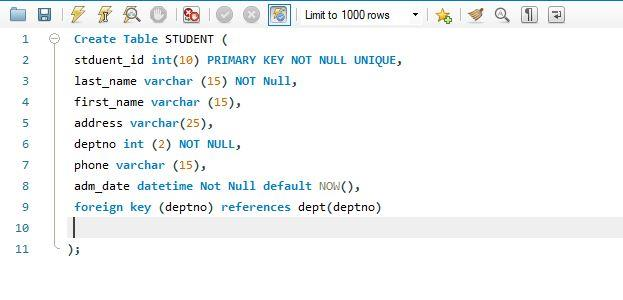


1. **Create EMP table with columns EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO. Make EMPNO as primary key and DEPTNO as foreign key.**

**Solution:**

1. **Create the STUDENT table based on the following table instance chart. Choose the appropriate data types and be sure to add integrity constraints.**

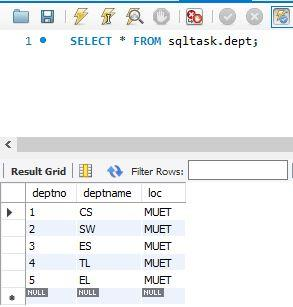


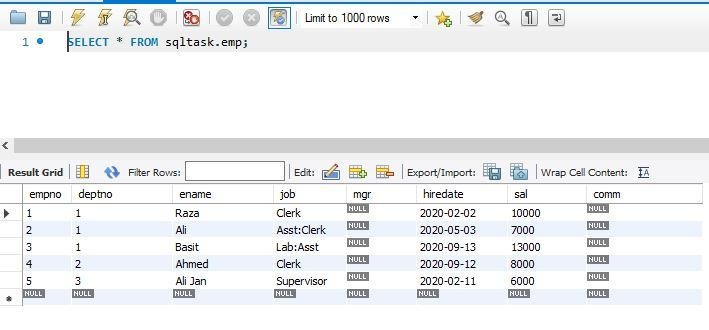
 **Solution:**

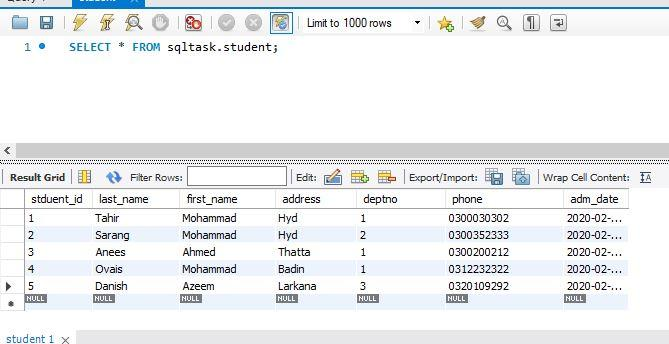
1. **Add at least 5 rows in each table by using constant values of your own choice.**

**Solution**:

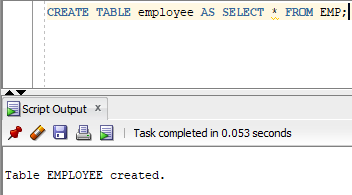
**Department**

****

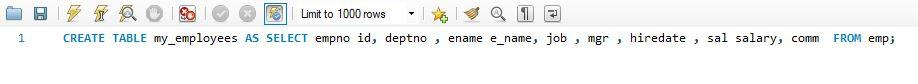
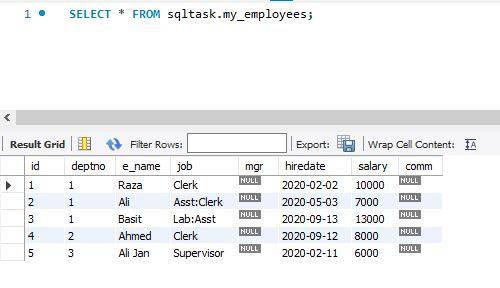
 **Emp Table:**

 **Student:**

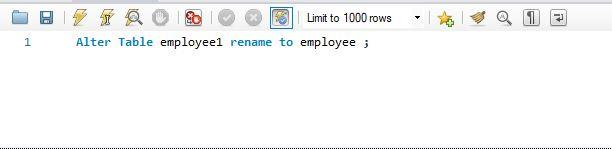
1. **Create the employee table based on the structure of EMP table.**

**Solution:**

1. **Create a table MY\_EMPLOYEE based on the structure of EMP table. Include the columns Empno, Ename, and Sal. Name them in new table as Id, Name, and Salary.**

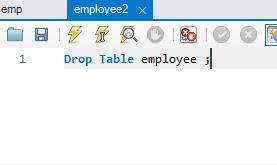
**Solution:**

**7. Rename employee1 table to Employee.**

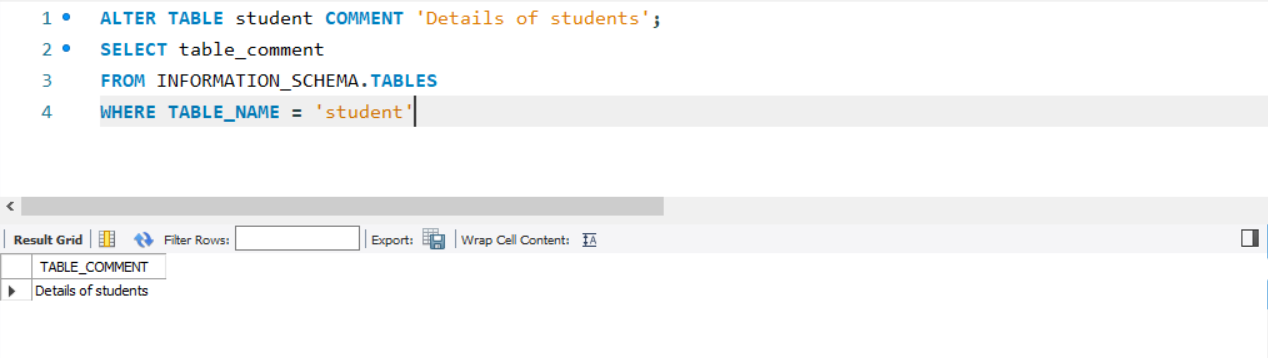
 **Solution:**

**8. Drop employee table.**

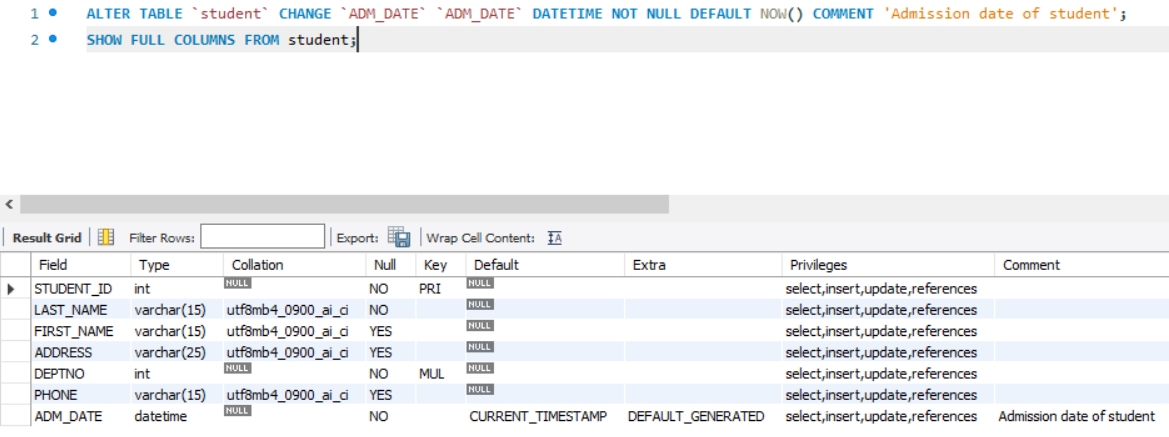
**Solution:**



**9. Add a comment to the student table describing the table.**

**Solution:**

**10. Add a comment on ADM\_DATE column as ‘Admission Date of student’.**

**Solution:**

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 4** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MySql / Oracle.**

**Introduction:**

**The SQL SELECT Statement**

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

**SELECT Syntax**

SELECT column1*,* column2 FROM table\_name;

Here, column1, column2 are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM table\_name;

## SELECT Column Example

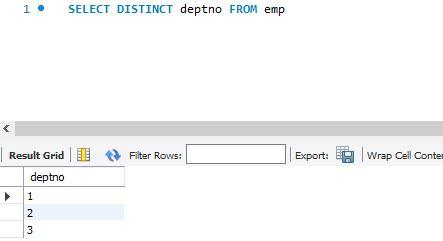
The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

### **Example**

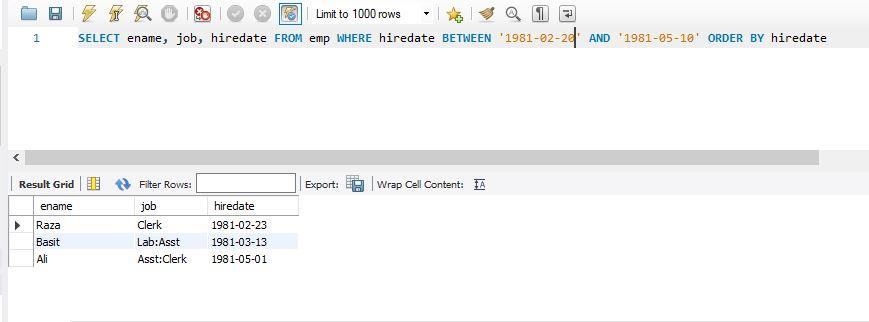
SELECT CustomerName, City FROM Customers;

**Lab Task**

**1. Create a query to display unique departments from EMP table.**

**Solution:**

**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.**

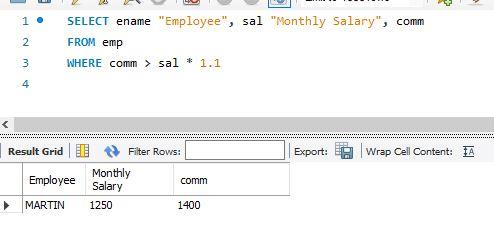
**Solution:**

**3. Display the names of all employees who have two ‘L’ in their name and are in department 30 or their manager is 7782.**

**Solution:**

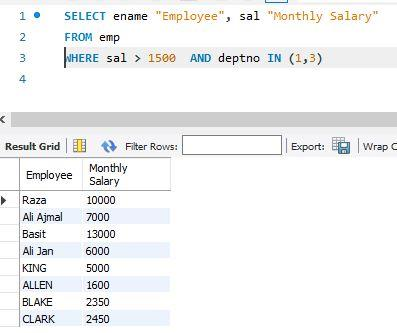


**4. Display the name, salary, and commission for all employees whose commission amount is greater than their salary increased by 10%.**

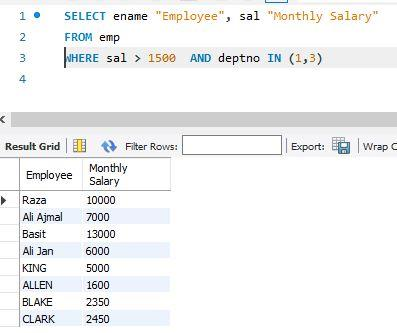
**Solution:**

**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.**

**Solution:**



**6. List out the employees whose name start with S and ends with H.**

**Solution:**

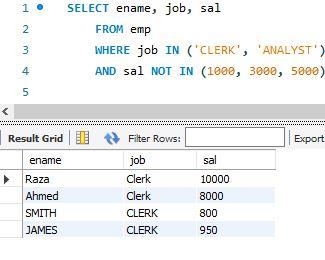
**7. Write a query that produces following for each employee: earns monthly but wants. Label the column Dream Salaries.**

**Solution:**



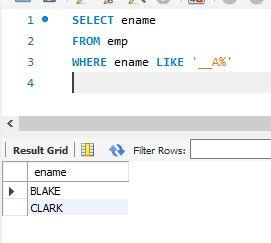
**8. Display the name job and salary for all employees whose job is clerk or analyst and their salary is not equal to 1000, 3000, or 5000.**

**Solution:**



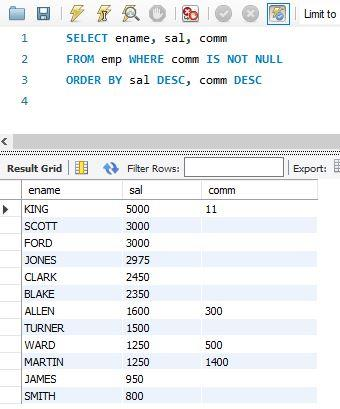
**9. Display the names of all employees where the third letter of their name is an A.**

**Solution:**



**10. Display name, salary and commission for all employees who earn commission. Sort the result in descending order of salary and commission.**

**Solution:**



|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 5** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MySql, Oracle.**

**Introduction:**

**Single Row functions** - Single row functions are the one who work on single row and return one output per row. For example, length and case conversion functions are single row functions.

**Single row functions**

Single row functions can be character functions, numeric functions, date functions, and conversion functions. Note that these functions are used to manipulate data items. These functions require one or more input arguments and operate on each row, thereby returning one output value for each row. Argument can be a column, literal or an expression. Single row functions can be used in SELECT statement, WHERE and ORDER BY clause. Single row functions can be -

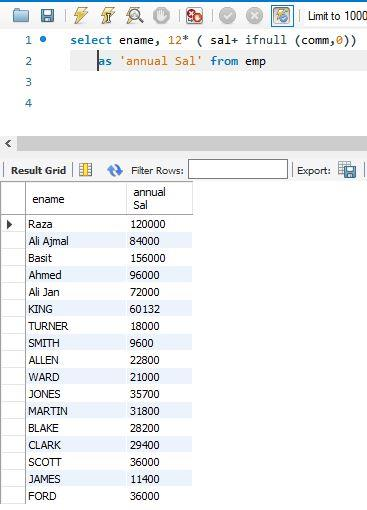
* **General functions** - Usually contains NULL handling functions. The functions under the category are NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.
* **Case Conversion functions** - Accepts character input and returns a character value. Functions under the category are UPPER, LOWER and INITCAP.
  + UPPER function converts a string to upper case.
  + LOWER function converts a string to lower case.
  + INITCAP function converts only the initial alphabets of a string to upper case.
* **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.
  + CONCAT function concatenates two string values.
  + LENGTH function returns the length of the input string.
  + SUBSTR function returns a portion of a string from a given start point to an end point.
  + INSTR function returns numeric position of a character or a string in a given string.
  + LPAD and RPAD functions pad the given string Upto a specific length with a given character.
  + TRIM function trims the string input from the start or end.
  + REPLACE function replaces characters from the input string with a given character.
* **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.
  + MONTHS\_BETWEEN function returns the count of months between the two dates.
  + ADD\_MONTHS function add 'n' number of months to an input date.
  + NEXT\_DAY function returns the next day of the date specified.
  + LAST\_DAY function returns last day of the month of the input date.
  + ROUND and TRUNC functions are used to round and truncates the date value.
* **Number functions** - Accepts numeric input and returns numeric values. Functions under the category are ROUND, TRUNC, and MOD.
  + ROUND and TRUNC functions are used to round and truncate the number value.
  + MOD is used to return the remainder of the division operation between two numbers.

**Lab Task**

**1. Display the employee name and their annual salary including commission amount. If the**

**Commission is null replace it with 0.**

**Task:**

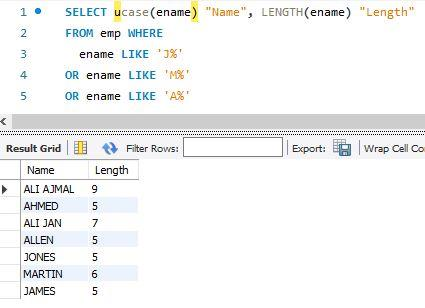


**2. Display Employee’s name with the first letter capitalized and all other letters lowercase and**

**the length of their name, for all employees whose name starts with J, A, or M. Give each column**

**appropriate names.**

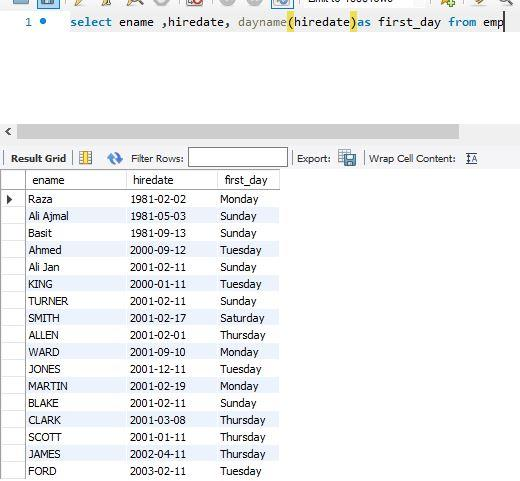
**Task:**



**3. Display the name, hiredate, and day of the week on which the employee started. Label the**

**Column First Day.**

**Task:**

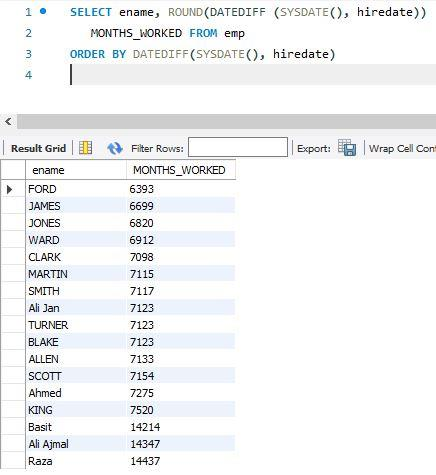


**4. For each employee, display the employee name and calculate the number of months between**

**today and the date the employee was hired. Label the column MONTHS\_WORKED. Order the**

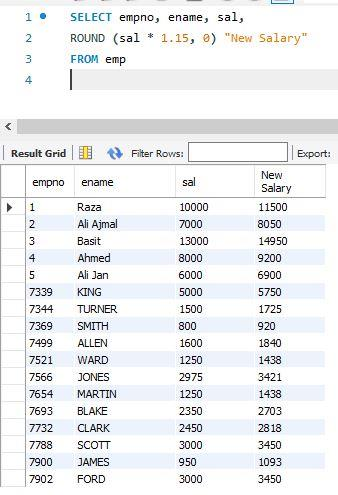
**result by the number of months employed. Round the number of months up to the closest whole**

**number.**

**Task:**

**5. Display the Employee number, name, salary, and salary increase by 15% expressed as a whole number. Label the column New Salary.**

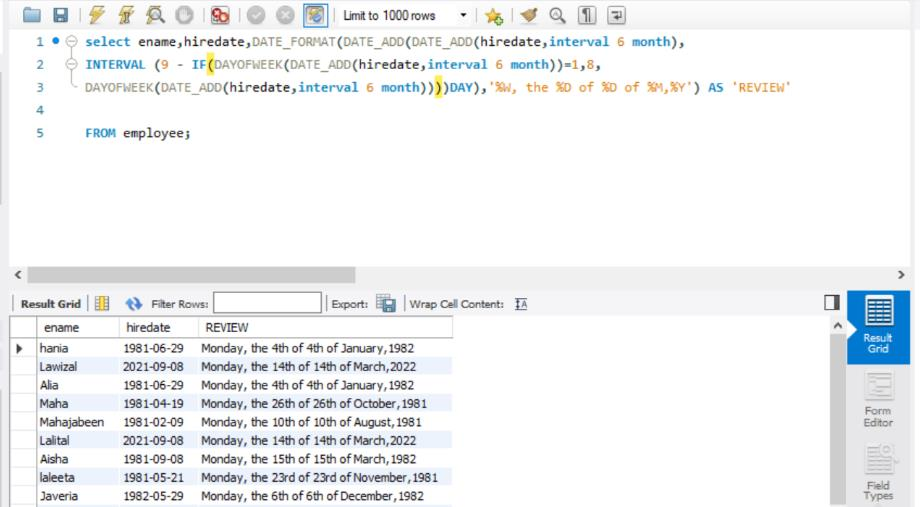
**Task:**



**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”.**

**Task:**



|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 6** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MySQL, Oracle**

**Introduction:**

**Group Function:** Group functions are mathematical functions to operate on sets of rows to give one result per set.

**Syntax**:

The general syntax for using Group functions is:

SELECT <column>, group\_function (column) FROM <table>

WHERE <condition> [GROUP BY <column>]

[ORDER BY <column>]

Note that the column on which the group function is applied must exist in the SELECT column list.

## Types of Group Function

Here are the different types of the Group function in SQL:

* **AVG, MIN, MAX, and SUM function:** you can use AVG, MIN, MAX and SUM for numeric data.

**Syntax**:

SELECT AVG(column\_name),MIN(column\_name),

SUM(column\_name), MAX(column\_name)

FROM table1 WHERE condition*;*

* **COUNT() Function:**
* COUNT(\*) function return the number of in a table;
* COUNT (expr) returns the number of rows with non-null values for the expr.
* COUNT (DISTINCT expr) returns the number of distinct non-null values of the expr.

**Syntax:**

SELECT COUNT(Column\_name)  
FROM table1  
WHERE condition*;*

* **NVL() Function:** The NVL() function forces Group functions to include the null values.

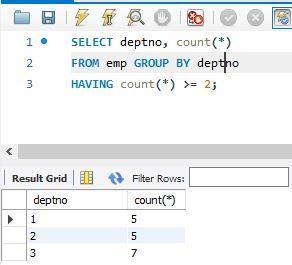
**Syntax:**

SELECT NVL(column\_name) FROM table11;

**Lab Task**

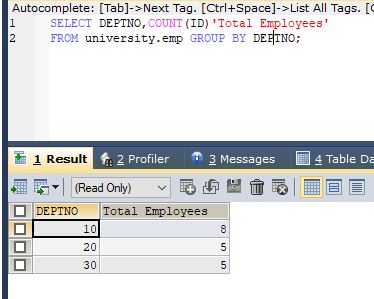
1. **List out the department numbers that have at least 4 employees.**

**Task:**



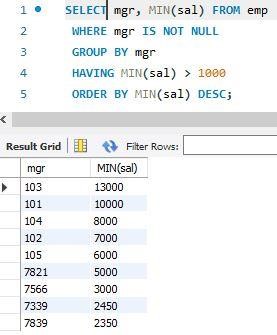
1. **Display the number of employees in each department.**

**Task:**

****

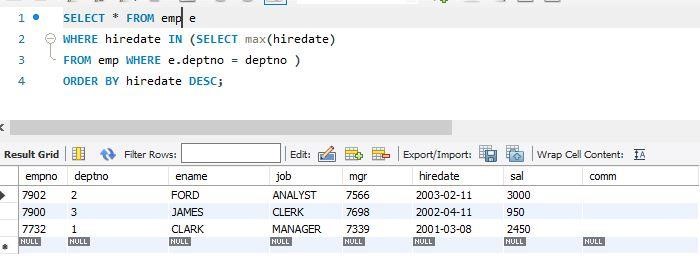
1. **Display the manager no and the salary of the lowest paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is less than 1000. Sort the output in descending order of salary.**

**Task:**

****

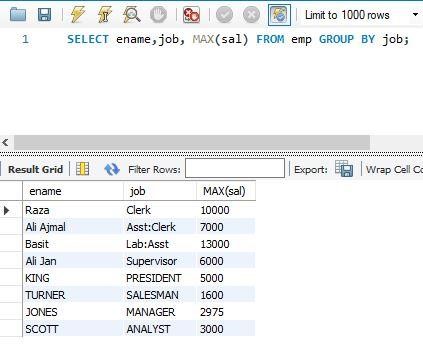
1. **Find the most recently hired employee in each department.**

**Task:**

****

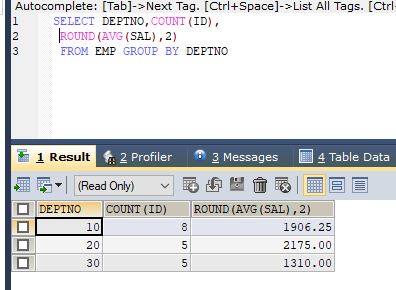
1. **List the highest salary paid for each job.**

**Task:**

****

1. **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.**

**Task:**

****

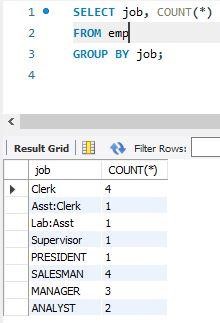
1. **Write a query that will display the difference between the highest and lowest salaries. Label the column DIFFERENCE.**

**Task:**

****

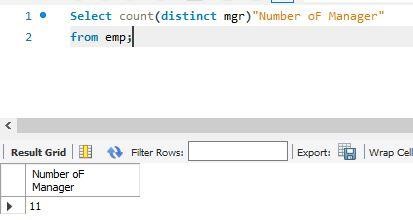
1. **Display the number of employees with same job.**

**Task:**

****

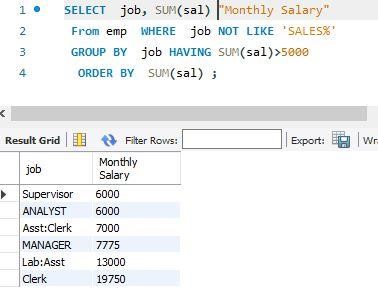
1. **Determine the number of managers without listing them. (Hint: Use MGR column)**

**Task:**

****

**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.**

**Task:**

****

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 7** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MySQL, Oracle**

**Introduction:**

**SQL Join:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let us look at a selection from the "Orders" table:



Then, look at a selection from the "Customers" table:

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

### **Example**

SELECT Orders.OrderID,Customers.CustomerName,Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

and it will produce something like this:



## Types of SQL Joins

Here are the different types of the JOINs in SQL:

* **SQL Inner Join:** The INNER JOIN keyword selects records that have matching values in both tables.

**Syntax**:

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name *=* table2.column\_name*;*

* **SQL Left Join:** The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side if there is no match.

**Syntax:**

SELECT column\_name(s)FROM table1LEFT JOIN table2ON table1.column\_name *=* table2.column\_name;

* **SQL Right join:** The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

**Syntax:**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

**Note:** In some databases RIGHT JOIN is called RIGHT OUTER JOIN.

* **SQL Full join:** The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**Note:** FULL OUTER JOIN can potentially return very large result-sets!

**Syntax:**

SELECT column\_name(s)FROM table1  
FULL OUTER JOIN table2ON table1.column\_name *=* table2.column\_nameWHERE condition;

* **SQL Self join:** A self-JOIN is a regular join, but the table is joined with itself.

**Syntax:**

SELECT column\_name(s)

FROM table1 T1, table1 T2

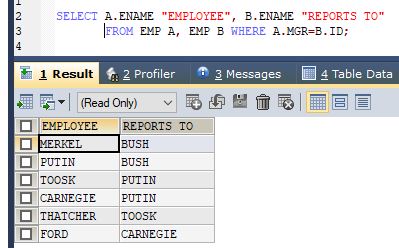
WHERE condition;

T1 and T2 are different table aliases for the same table.

**Lab Task**

**1. List the name of the employees with the name of their immediate higher authority.**

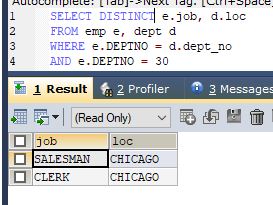
**Task:**

****

**2. Create a unique listing of all jobs that are in department 30. Include the location of**

**department 30 in the output.**

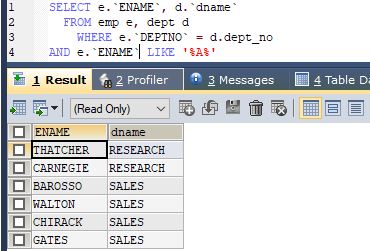
**Task:**

****

**3. Display the employee name and department name for all employees who have an A in**

**their name.**

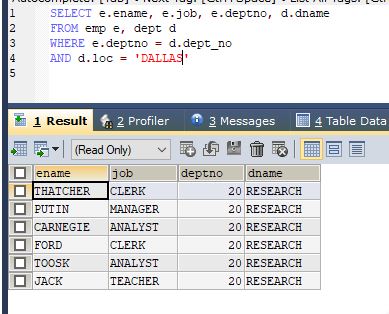
**Task:**

****

**4. Display the employee name and department name for all employees who work in**

**DALLAS.**

**Task:**

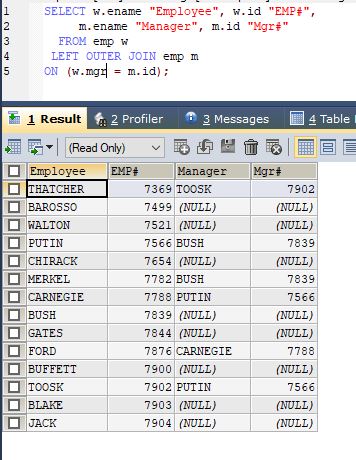
****

**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**

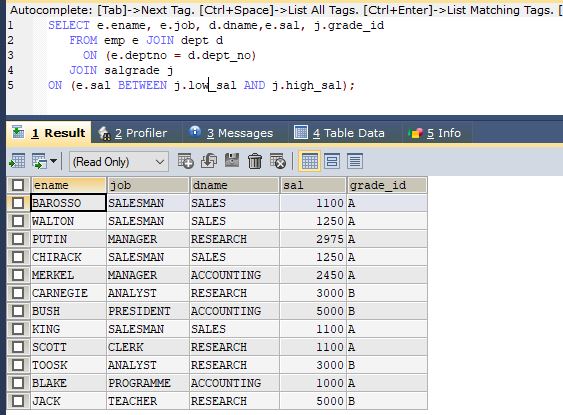
**columns employee, Emp#, Manager, and Mgr#, respectively.**

**Task:**

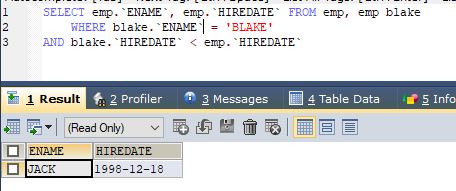
****

**6. Create a query that will display the name, job, department name, salary, grade for all**

**employees.**

**Task:**

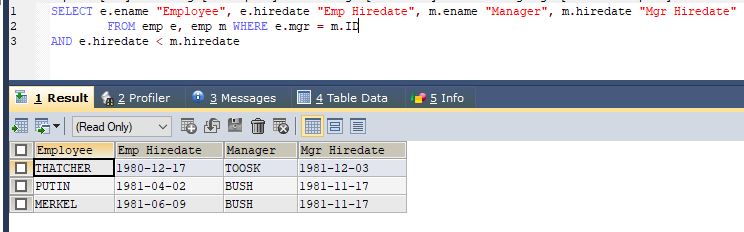
**7. Create a query to display the name and hire date of any employee hired after employee BLAKE.**

**Task:**

**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.**

**Task:**

****

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 8** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

# Objective: To study SQL sub-queries.

**Tools:** MySql

## What is subquery in SQL?

A subquery is a SQL query nested inside a larger query.

* A subquery may occur in :
  + - A SELECT clause
  + - A FROM clause
  + - A WHERE clause
* The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.
* A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
* You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.
* A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.
* The inner query executes first before its parent query so that the results of an inner query can be passed to the outer query.

You can use a subquery in a SELECT, INSERT, DELETE, or UPDATE statement to perform the following tasks:

* Compare an expression to the result of the query.
* Determine if an expression is included in the results of the query.
* Check whether the query selects any rows.

**Syntax :**

SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

WHERE column\_name OPERATOR

(SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

[WHERE])

* The subquery (inner query) executes once before the main query (outer query) executes.
* The main query (outer query) use the subquery result.

**Lab Task**

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 Management Systems (4th Semester) 18CS**  **Lab Experiment 09** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

**Objective: To understand Indexes, Views & Sequences.**

**Tools: MySQL/Oracle.**

**Introduction:**

**INDEXES:** A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns. The users cannot see the indexes, they are just used to speed up queries and will be used by the Database Search Engine to locate records very fast.

* The INSERT and UPDATE statements take more time on tables having indexes, whereas the SELECT statements become fast on those tables.
* The reason is that while doing insert or update, a database needs to insert or update the index values as well.

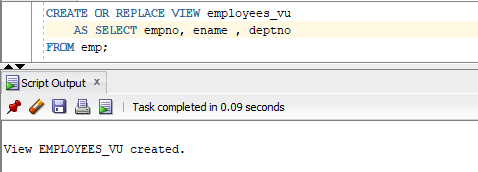
**VIEWS:** In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

* You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
* Note that a view does not physically store the data.

**SEQUENCE:** A sequence is a list of [integers](https://www.mysqltutorial.org/mysql-int/) generated in the ascending order**.** Many applications need sequences to generate unique numbers mainly for identification

**Lab Task**

**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.**

**Task**

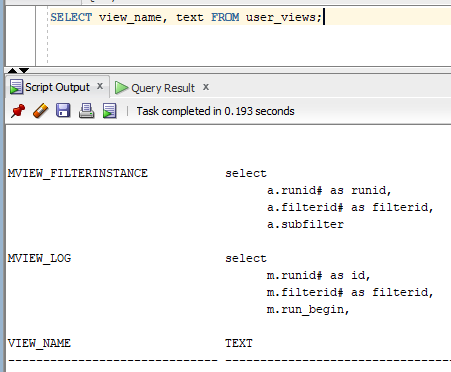
**2. Display the contents of the EMP\_VU view.**

**Task**

****

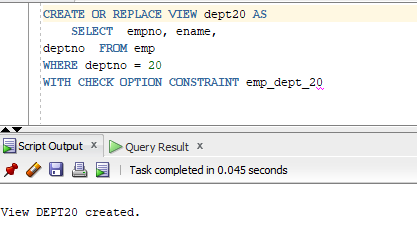
**3. Select the view name and text from the data dictionary USER\_views.**

**Task**

****

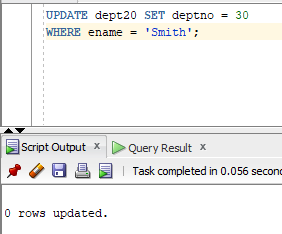
**4. Create a view named DEPT\_VU\_20 that contains the employee number, employee name and department number for all employees in department 20. Label the view column Employee\_Id, Employee, and Department\_Id. Do not allow an employee to be reassigned to another department through the view.**

**Task**

****

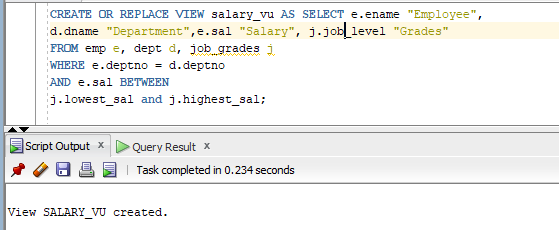
**5. Attempt to reassign Smith to department 30.**

**Task**

****

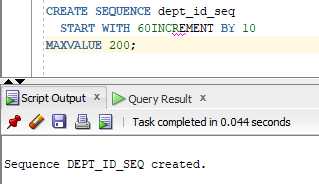
**6. Create a view Salary\_vu based on the employee name, department name, salary and salary grade for all employees. Label the columns Employee, Department, Salary and Grade.**

**Task**

****

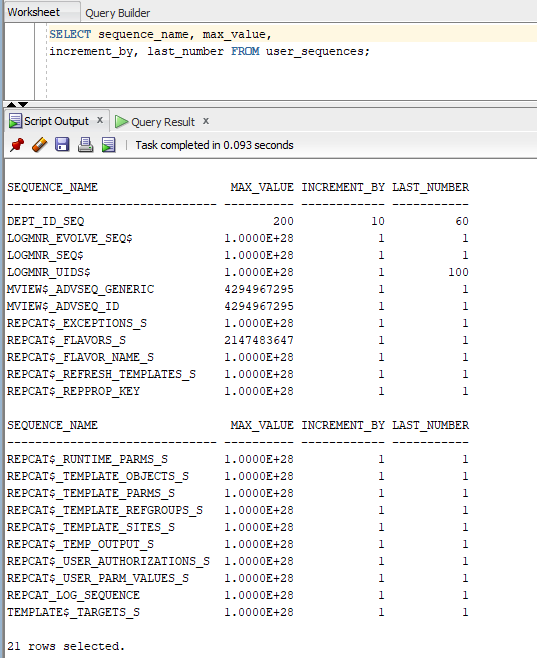
**7. Create a sequence to be used with primary key column of the department table. The sequence should start at 60 and have a maximum value of 200. Have your sequence increment by ten numbers. Name the sequence Dept\_Id\_Seq.**

**Task**

****

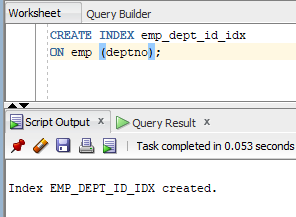
**8. Display the following information about your sequences: sequence name, max value, increment size, and last number.**

**Task**

****

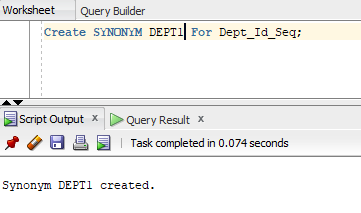
**9. Create a non-unique index on the foreign key column in the employee table.**

**Task**

****

**10. Create a synonym for Dept\_Id\_Seq.**

**Task**

****

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 10** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

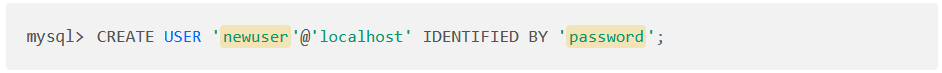
**Objective: Creating database users and controlling user access.**

**Tools: MySQL/Oracle.**

**Introduction:**

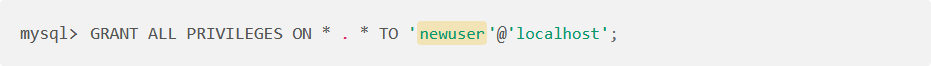
**Create a New User**

MySQL has a CREATE USER statement that lets you create a new user on your database. If you're working with a new database, you might only have a root user set up. To create users, you first need to log in with the root user or another user account that has permissions to create users.



At this point *newuser* has no permissions to do anything with the databases. In fact, even if *newuser* tries to login (with the password, *password*), they will not be able to reach the MySQL shell.

Therefore, the first thing to do is to provide the user with access to the information they will need.



**How to Grant Different User Permissions**

There are numerous table and column level privileges that you can grant. You can also grant users the privileges to create other users or revoke other user privileges. The permissions you grant to a user should depend on what level of access you think they should have. The rule for security permissions is to give a user the least amount of privileges they need to accomplish normal tasks. For instance, you shouldn't grant all permissions to a standard user when they only need to run SELECT statements on a Customer table.

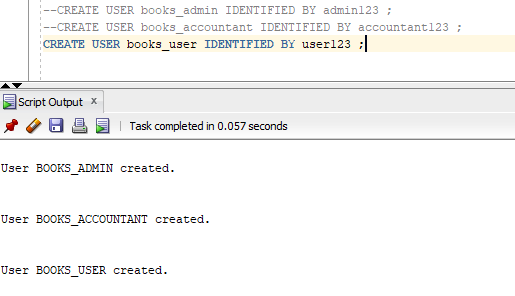
Here is a short list of other common possible permissions:

* ALL PRIVILEGES- as we saw previously, this would allow a MySQL user full access to a designated database (or if no database is selected, global access across the system)
* CREATE- allows them to create new tables or databases
* DROP- allows them to them to delete tables or databases
* DELETE- allows them to delete rows from tables
* INSERT- allows them to insert rows into tables
* SELECT- allows them to use the SELECT command to read through databases
* UPDATE- allow them to update table rows
* GRANT OPTION- allows them to grant or remove other users’ privilege

This practical is an overview of managing user accounts and passwords. Keeping track of users and their permissions greatly reduces the chance that your database goes hacked without notice. By managing users, you keep granular control over your data and protect it from unauthorized access.

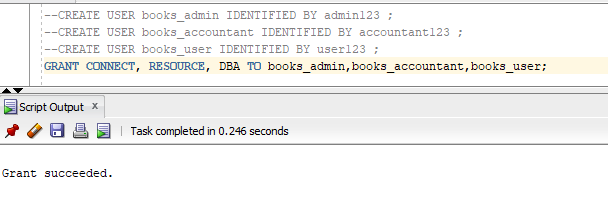
**Lab Task**

**1. For a book library, create 3 users with names books\_admin, books\_accountant, books\_user and assign passwords of your own choice.**

**Task:**

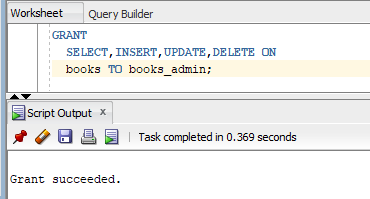
**2. Assign DBA, RESOURCE and CONNECT roles to books\_admin, books\_accountant, books\_user, respectively.**

**Task:**

****

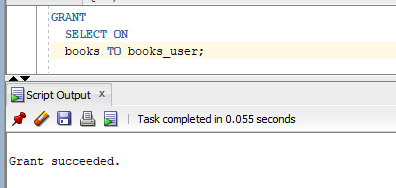
**3. Assign select, update, delete, and insert object privileges on books table to books\_admin and books\_accountant users.**

**Task:**

****

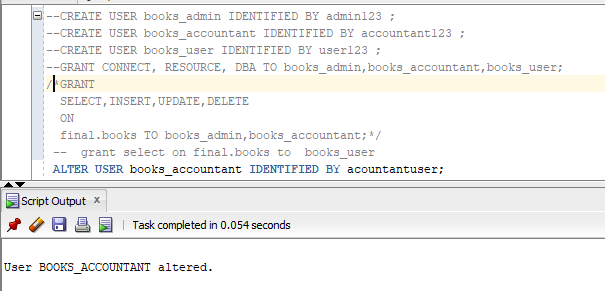
**4. Assign select object privilege on books table to books\_user user.**

**Task:**

****

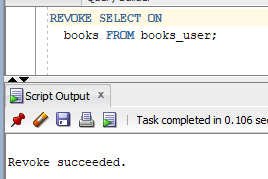
**5. Change the password of user books\_accountant.**

**Task:**

****

**6. Revoke the assigned object privilege from books\_user.**

**Task:**

****

**7. What privilege should a user be given to log into the oracle server? Is this a system or object privilege?**

**Answer:** The CREATE SESSION system privilege.

**8. What privilege should a user be given to create tables and views?**

**Answer:** The CREATE TABLE privilege.

**9. If you create a table, who can pass along privileges to other users on your table?**

**Answer:** You can, or anyone you have given those privileges to by using the WITH GRANT OPTION.

**10. You are the DBA. You are creating many users who require the same system privileges. What would you use to make your job easier?**

**Answer:** Create a role containing the system privileges and grant the role to the users.

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 11** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MySql, Oracle.**

**Introduction:**

**PL/SQL Block Structure**

In PL/SQL, as in most other procedural languages, the smallest meaningful grouping of code is known as a block. A block is a unit of code that provides execution and scoping boundaries for variable declarations and exception handling. PL/SQL allows you to create anonymous blocks (blocks of code that have no name) and named blocks, which may be packages, procedures, functions, triggers, or object types..

**A PL/SQL block has up to four different sections, only one of which is mandatory:**

**Header**

Used only for named blocks. The header determines the way the named block or program must be called. Optional.

**Declaration section**

Identifies variables, cursors, and subblocks that are referenced in the execution and exception sections. Optional.

**Execution section**

Statements the PL/SQL runtime engine will execute at runtime. Mandatory.

**Exception section**

Handles exceptions to normal processing (warnings and error conditions). Optional

**Basic Syntax structure**

**Declare**

Declaration of variable, constants

**Begin**

Execute statements in pl/sql

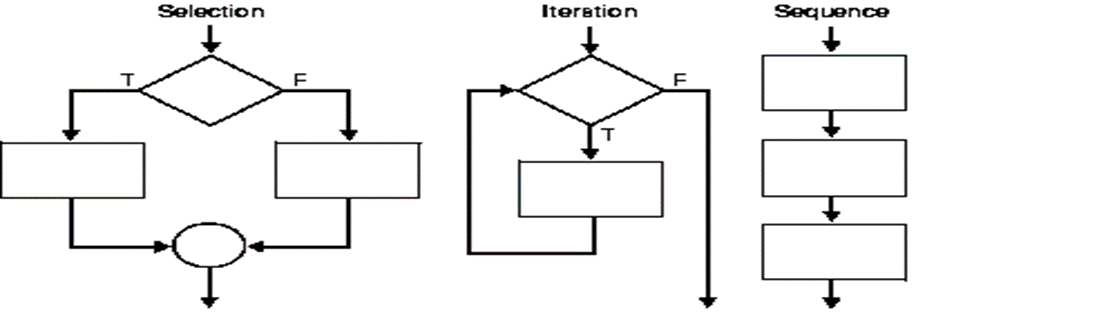
**Exception**

Exception handlers in pl/sql

**END;**

**PL/SQL Control Structures**

The selection structure tests a condition, then executes one sequence of statements instead of another, depending on whether the condition is true or false. A condition is any variable or expression that returns a Boolean value (TRUE or FALSE). The iteration structure executes a sequence of statements repeatedly as long as a condition holds true. The sequence structure simply executes a sequence of statements in the order in which they occur.

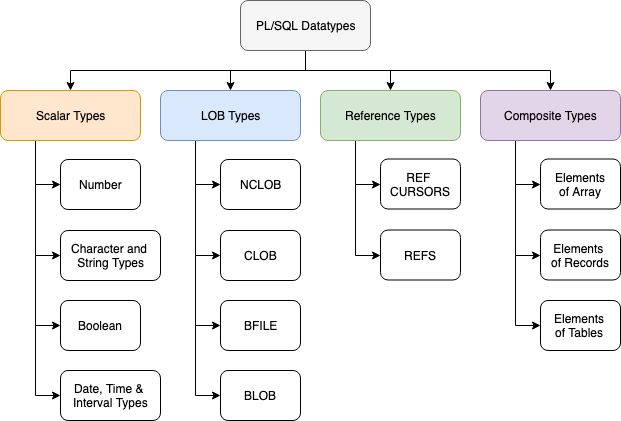


**PL/SQL Data Types.**

PL/SQL datatypes are not just limited to writing SQL queries but they are used in the PL/SQL block as well, just like any other programming language.

Provising a datatype specifies how any data will be stored and processed by Oracle when any PL/SQL code block is executed.

Datatype defines the type of data being used, whether it is a number or a word (string) or a single character etc. Following datatypes can be used in PL/SQL depending upon the type of data required:



**So we have 4 broader categories of datatypes and they are:**

**1. Scalar Types:** These are basic datatypes which generally holds a single value like a number or a string of characters. Scalar types have 4 different categories which are listed in the diagram above, namely Number Types, Character and String, Boolean Types and Date and Time etc.

2. **LOB Types**: This datatype deals with large objects and is used to specify location of these large objects like text files, images etc which are generally not stored outside the database.

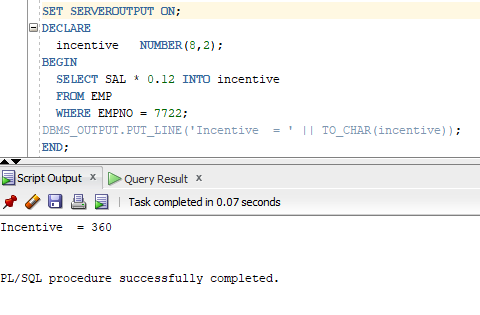
3. **Reference Types:** This datatype is used to hold pointer values which generally stores address of other program items.

4. **Composite Types**: Last but not the least, as the name suggests this type of data is a composition of individual data which can be manipulated/processed separatel as well.

**LAB TASK**

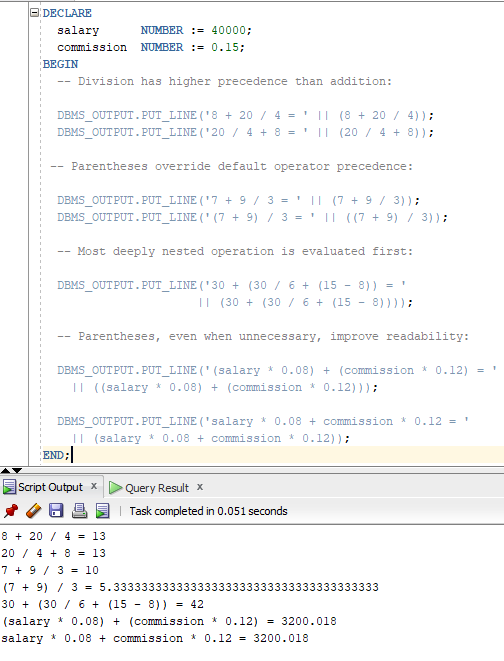
1. **Write a PL/SQL block to calculate the annual salary of an employee whose ID is 7722.**

**Task:**

****

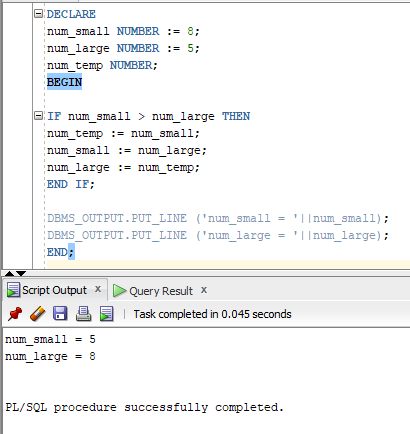
1. **Write a PL/SQL block to show the operator precedence and parentheses in 5 or more complex expressions.**

**Task:**

****

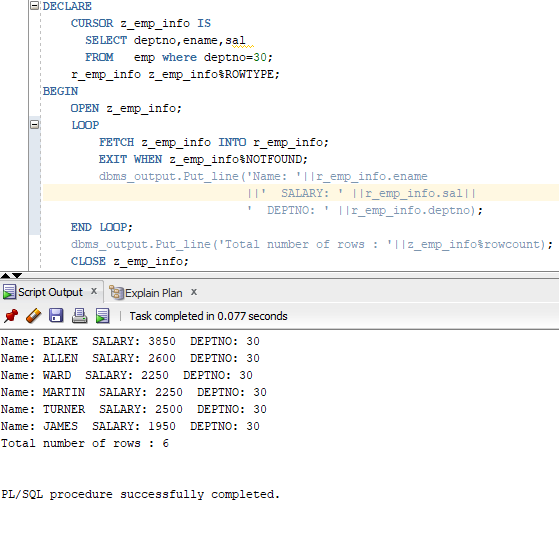
1. **Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num\_small variable and large number will store in num\_large variable.**

**Task:**

****

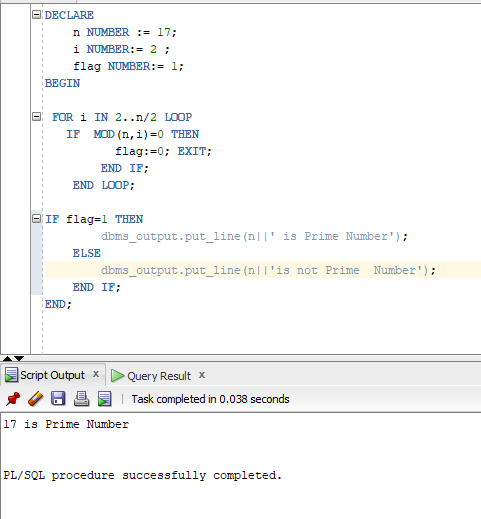
1. **Write a PL/SQL program to count number of employees in department 30 and check whether this department have any vacancies or not. There are total 45 posts in this department.**

**Task:**

****

1. **Write a program in PL/SQL to check whether a number is prime or not using for loop.**

**Task:**



|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 12** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

**Objective**: **To explore Exception Handling in PL/SQL.**

**Tools: MySql, Oracle.**

**Introduction:**

An exception occurs when the PL/SQL engine encounters an instruction which it cannot execute due to an error that occurs at run-time. These errors will not be captured at the time of compilation and hence these needed to handle only at the run-time.

For example: if PL/SQL engine receives an instruction to divide any number by '0', then the PL/SQL engine will throw it as an exception. The exception is only raised at the run-time by the PL/SQL engine.

Exceptions will stop the program from executing further, so to avoid such condition, they need to be captured and handled separately. This process is called as Exception-Handling, in which the programmer handles the exception that can occur at the run time.

## Exception-Handling Syntax:

## Exceptions are handled at the block, level, i.e., once if any exception occurs in any block then the control will come out of execution part of that block. The exception will then be handled at the exception handling part of that block.

## 

## Types of Exception:

1. Predefined Exceptions
2. User-defined Exception

## Predefined Exceptions: Oracle has predefined some common exception. These exceptions have a unique exception name and error number. These exceptions are already defined in the 'STANDARD' package in Oracle. In code, we can directly use these predefined exception name to handle them.

## User-defined Exception: In Oracle, other than the above-predefined exceptions, the programmer can create their own exception and handle them. They can be created at a subprogram level in the declaration part. These exceptions are visible only in that subprogram. The exception that is defined in the package specification is public exception, and it is visible wherever the package is accessible.

DECLARE

<exception\_name> EXCEPTION;

BEGIN

<Execution block>

EXCEPTION

WHEN <exception\_name> THEN

<Handler>

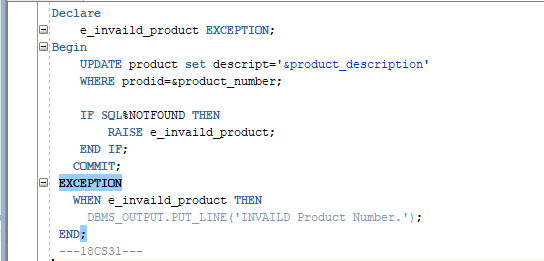
END;

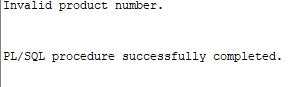
* In the above syntax, the variable 'exception\_name' is defined as 'EXCEPTION' type.
* This can be used as in a similar way as a predefined exception.

**Lab Task**

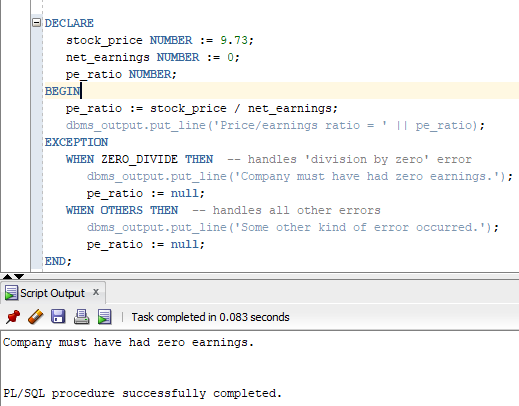
**1. Write a PL/SQL block that updates description of a product and raises a user-defined exception when that product is not found.**

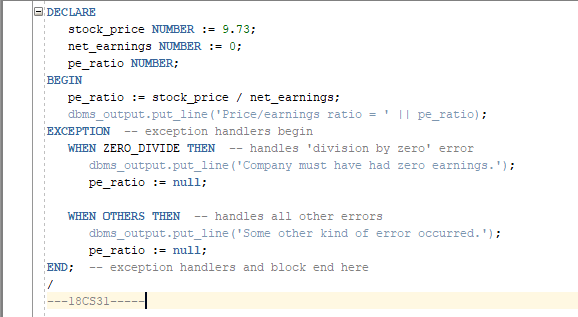
**Task:**

****

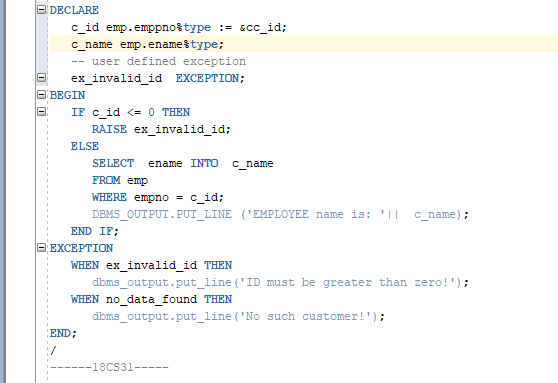
**Output:**

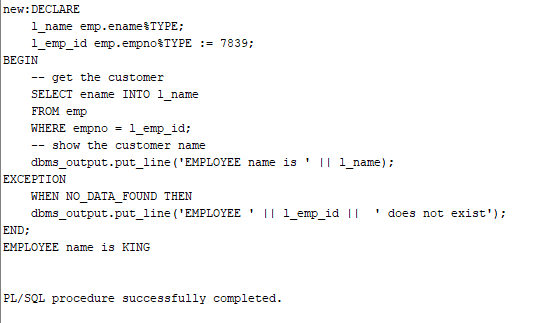
**2. Write a simple PL/SQL code block, to demonstrate the use of Named Exception Handler for division by zero error.**

**Task:**



**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.**

**Task:**

**Output:**

|  |  |
| --- | --- |
| **A picture containing logo  Description automatically generated** | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 13** |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools: MYSQL** Oracle.

**Introduction:**

**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

**Implicit cursors**: Whenever Oracle executes an SQL statement such as [SELECT INTO](https://www.oracletutorial.com/plsql-tutorial/plsql-select-into/), [INSERT](https://www.oracletutorial.com/oracle-basics/oracle-insert-into-select/), [UPDATE](https://www.oracletutorial.com/oracle-basics/oracle-update/), and [DELETE](https://www.oracletutorial.com/oracle-basics/oracle-delete/), it automatically creates an implicit cursor.

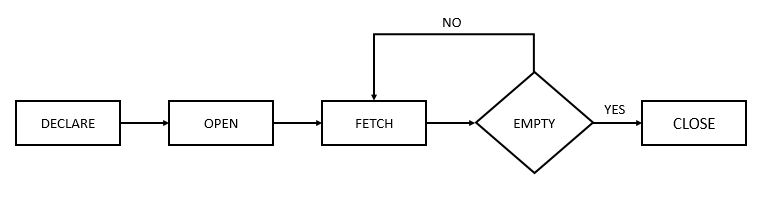
Oracle internally manages the whole execution cycle of implicit cursors and reveals only the cursor’s information and statuses such as SQL%ROWCOUNT, SQL%ISOPEN, SQL%FOUND, and SQL%NOTFOUND.

The implicit cursor is not elegant when the query returns zero or multiple rows which cause NO\_DATA\_FOUND or TOO\_MANY\_ROWS exception respectively.

**Explicit cursors**: An explicit cursor is an [SELECT](https://www.oracletutorial.com/oracle-basics/oracle-select/) statement declared explicitly in the declaration section of the current block or a package specification.

For an explicit cursor, you have control over its execution cycle from OPEN, FETCH, and CLOSE.

Oracle defines an execution cycle that executes an SQL statement and associates a cursor with it. The following illustration shows the execution cycle of an explicit cursor



### **Explicit Cursor Attributes:**

A cursor has four attributes to which you can reference in the following format: cursor\_name%attribute (where cursor\_name is the name of the explicit cursor)

1. **%ISOPEN:**

This attribute is TRUE if the cursor is open or FALSE if it is not.

1. **%FOUND:**

This attribute has four values:

* NULL before the first fetch
* TRUE if a record was fetched successfully
* FALSE if no row returned
* INVALID\_CURSOR if the cursor is not opened

1. **%NOTFOUND:**

This attribute has four values:

* NULL before the first fetch
* FALSE if a record was fetched successfully
* TRUE if no row returned
* INVALID\_CURSOR if the cursor is not opened

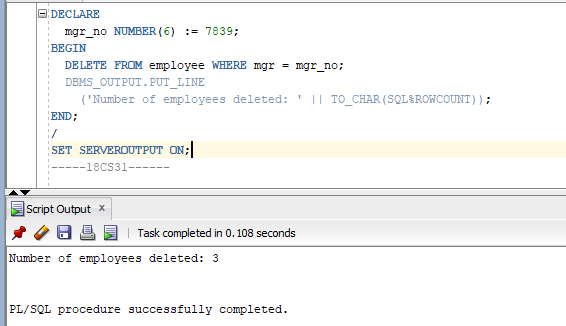
1. **%ROWCOUNT:**

The %ROWCOUNT attribute returns the number of rows fetched from the cursor. If the cursor is not opened, this attribute returns INVALID\_CURSOR.

**Lab Task**

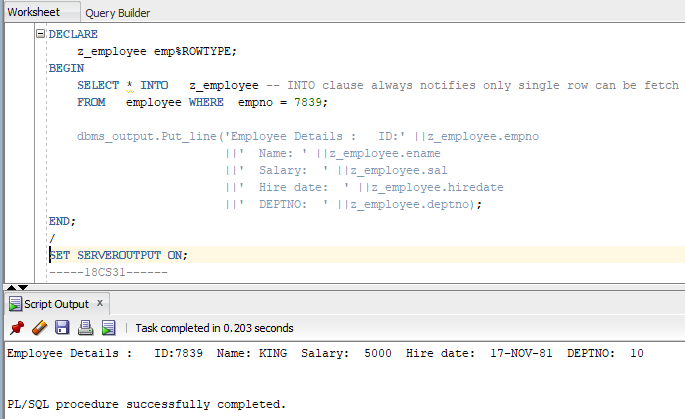
1. **Write a program in PL/SQL to find the number of rows effected using SQL%ROWCOUNT attributes of an implicit cursor.**

**Task:**

****

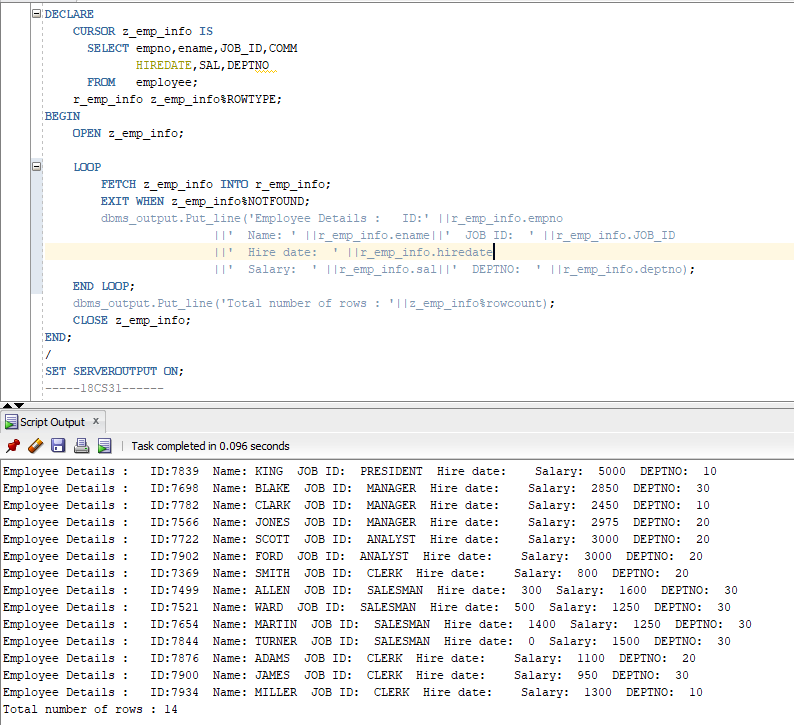
1. **Write a program in PL/SQL to display detail information for the employee of ID 7839 from the employees’ table.**

**Task:**

****

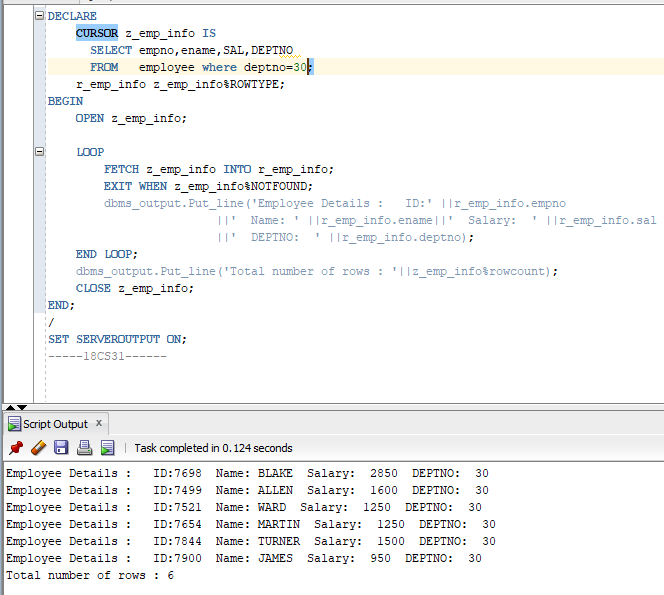
1. **Write a program in PL/SQL to display detail information of all employees from employees table using explicit cursor.**

**Task:**

****

1. **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.**

**Task:**



|  |  |
| --- | --- |
| A picture containing logo  Description automatically generated | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 14** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll No:** |  | **Date of Conduct:** |  |
| **SubmissionDate:** |  | **Grade Obtained:** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

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

**Tools:** **MySql/oracle**.

**Introduction:**

**Stored Functions and Procedures in PL/SQL**

Stored procedures and functions (subprograms) can be compiled and stored in an Oracle Database, ready to be executed. Once compiled, it is a schema object known as a stored procedure or stored function, which can be referenced or called any number of times by multiple applications connected to Oracle Database. Both stored procedures and functions can accept parameters when they are executed (called). To execute a stored procedure or function, you only need to include its object name.

**Creating a Stored Procedure or Function**

Whenever a block of code for stored procedure or function is written it is then,they are automatically compiled by the oracle engine. During compilation if any error occurs, we get a message on the screen saying that the procedure or function is created with compilation errors but actual error is not displayed.

In order to find out the compilation errors following statement can be executed:

SELECT \* from USER\_ERRORS;

Once it is compiled, it is then stored by the oracle engine in the database as a database object.

Stored Procedure or function's block of code in PL/SQL is made up of the following three sections:

* **Declarative section:** In this section, variables, constants, cursor or exceptions that are going to be used by procedure or function are declared.
* **Executable section:** In this section, the definition of procedure or function created is written. This section also contains the SQL or PL/SQL statements assigning values, controlling execution and manipulating data.
* **Exception Handling section:** In this section, the expected exceptions are written which may arise during execution of code written in executable part. This section is optional.

**Syntax for creating Stored Procedure**

Below we have the basic syntax for creating a stored procedure in oracle:

CREATE OR REPLACE PROCEDURE <procedure\_name>

(<variable\_name>IN/OUT/IN OUT <datatype>,

<variable\_name>IN/OUT/IN OUT <datatype>,...) IS/AS

variable/constant declaration;

BEGIN

-- PL/SQL subprogram body;

EXCEPTION

-- Exception Handling block ;

END <procedure\_name>;

Let's understand the above code,

* procedure\_name is for procedure's name and variable\_name is the variable name for variable used in the stored procedure.
* CREATE or REPLACE PROCEDURE is a keyword used for specifying the name of the procedure to be created.
* BEGIN, EXCEPTION and END are keywords used to indicate different sections of the procedure being created.
* IN/OUT/IN OUT are parameter modes.
* IN mode refers to READ ONLY mode which is used for a variable by which it will accept the value from the user. It is the default parameter mode.
* OUT mode refers to WRITE ONLY mode which is used for a variable that will return the value to the user.
* IN OUT mode refers to READ AND WRITE mode which is used for a variable that will either accept a value from the user or it will return the value to the user.
* At the end, <procedure\_name> is optional to write, you can simply use END statement to end the procedure definition.

**Syntax for creating Functions in PL/SQL**

Now that we know how to create stored procedures and how to use them in another PL/SQL code block, it's time to understand how to create Functions in PL/SQL.

CREATE OR REPLACE FUNCTION <function\_name>

(<variable\_name> IN <datatype>,

<variable\_name> IN <datatype>,...)

RETURN <datatype> IS/AS

variable/constant declaration;

BEGIN

-- PL/SQL subprogram body;

EXCEPTION

-- Exception Handling block ;

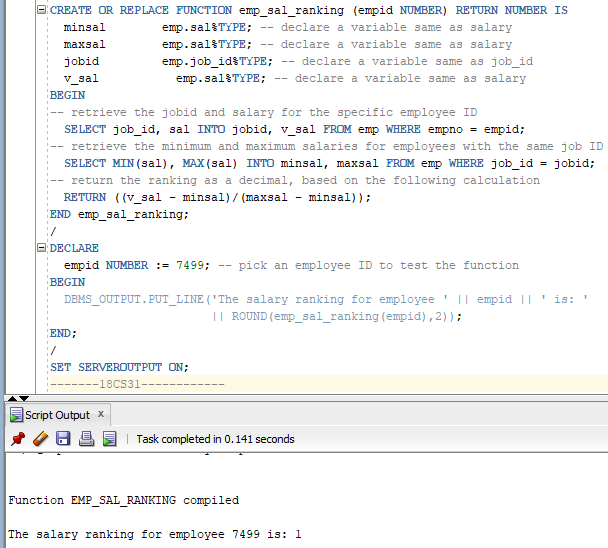
END <function\_name>;

Let's understand the above code:

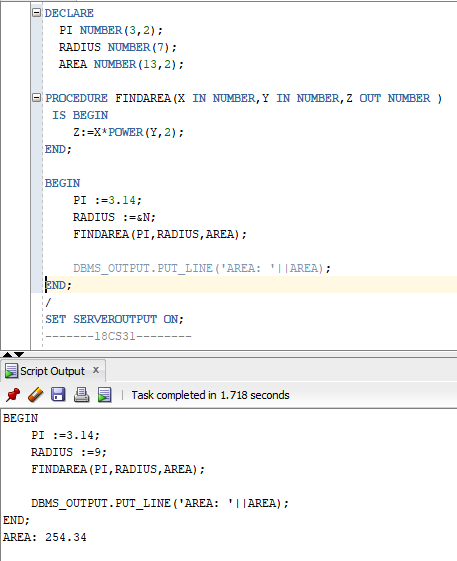
* Function\_name is for defining function's name and variable\_name is the variable name for variable used in the function.
* CREATE or REPLACE FUNCTION is a keyword used for specifying the name of the function to be created.
* IN mode refers to READ ONLY mode which is used for a variable by which it will accept the value from the user. It is the default parameter mode.
* RETURN is a keyword followed by a datatype specifying the datatype of a value that the function will return.

**LAB TASK**

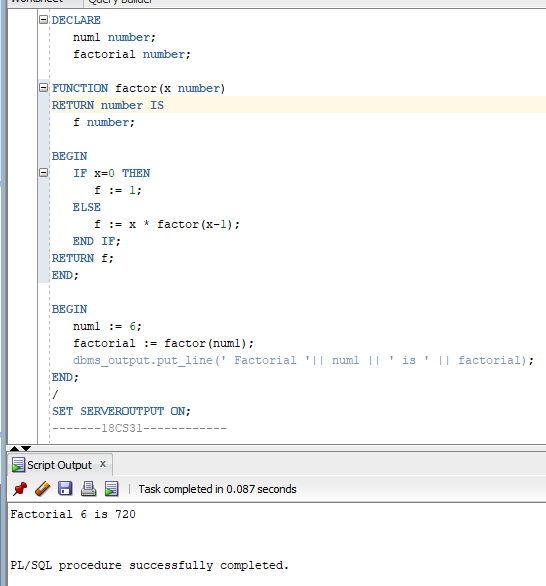
**1. Create a stored function to calculate the salary ranking of the employee based on the  
current minimum and maximum salaries for employees in the same job category. (Hint:  
salary ranking = (empsal-minsal)/(Maxsal-Minsal)).**

**Task:**

**2. Write a stored procedure that accepts radius of circle and displays area of the circle.**

**Task:**

**3. Write a stored function that accepts a number and returns its factorial.**

**Task:**

|  |  |
| --- | --- |
| A picture containing logo  Description automatically generated | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 15** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll No:** |  | **Date of Conduct:** |  |
| **SubmissionDate:** |  | **Grade Obtained:** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

**Objective:** **To understand the use of Triggers in PL/SQL.**

**Tools:** **MySql/oracle**.

**Introduction:**

**Triggers:** Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

* A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
* A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
* A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP,

Or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

**Benefits of Triggers**

Triggers can be written for the following purposes −

* Generating some derived column values automatically
* Enforcing referential integrity
* Event logging and storing information on table access
* Auditing
* Synchronous replication of tables
* Imposing security authorizations
* Preventing invalid transactions

**Creating Triggers**

**The syntax for creating a trigger is −**

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

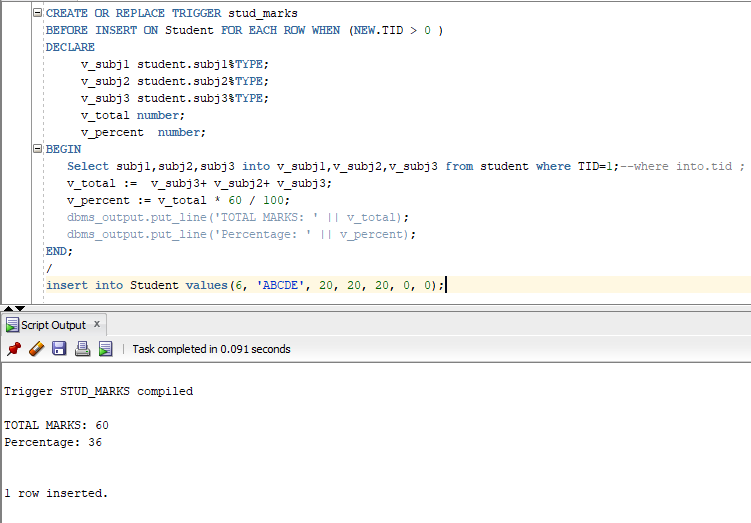
END;

Where,

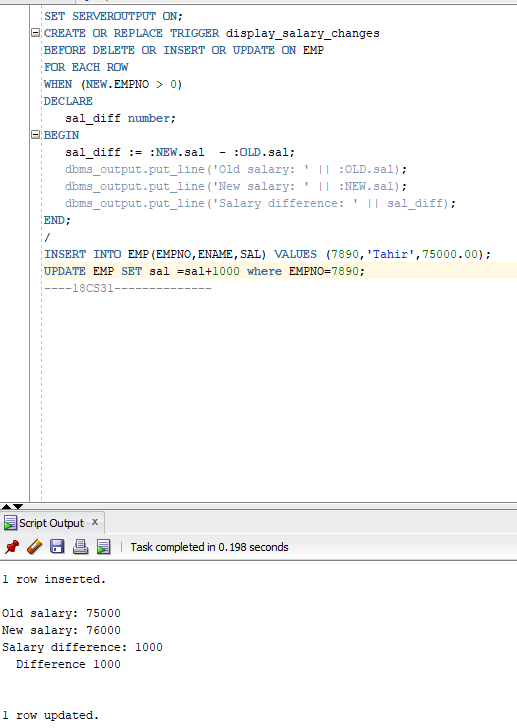
* CREATE [OR REPLACE] TRIGGER trigger\_name − Creates or replaces an existing trigger with the *trigger\_name*.
* {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
* [OF col\_name] − This specifies the column name that will be updated.
* [ON table\_name] − This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
* [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition) − this provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

**LAB TASK**

**1. Consider Student table, in which student’s marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is inserted.**

**Task:**

**2. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger should display the salary difference between the old values and new values.**

**Task:**