**Profile Report**

## CANDIDATE DETAILS

Lerner (GRTLN1000)

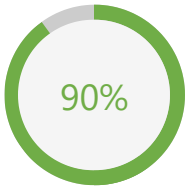
learner@maantt.com

Group Name : ADMIN

## OVERALL COMPETENCY

**Pass**

*Pass Percentage : 50*




## ASSESSMENT DETAILS

Assessment Name : Testing pivot, function, procedure, sub query etc.,

Test Marks : 90.00

Total Questions : 1

Penalty : No

Partial Marks : Yes

## ATTEMPT DETAILS

Attempt No : 1

Started At : 14-05-2019 2:44PM

Submitted At : 14-05-2019 2:54PM

Time Taken : 10 Mins

## CANDIDATE SCORE

Questions Answered : 90.00

Questions Not Answered : 0

Marks for Correct Answer : 0.00

Partial Marks : 90.00

Your Score : 90.00

## ANSWER SUMMARY


## TO RETURN EMPLOYEE HISTORY DETAILS

Write a query to return employee history details like LastName , FirstName , EmployeeID , Department and Rate.

**Design Rules :**

1. Use Joins

2. Rename the column 'Name' from Department table as 'Department' in the resultset

3. Return only employees whose FirstName starts with letter 'P'

**Tables:**

Department, EmployeeDepartmentHistory, EmployeePayHistory, Employee, Contact

Note : Each employee can have more  than one department mapped in the EmployeeDepartmentHistory.

**Sample Output:**

| LASTNAME | FIRSTNAME | EMPLOYEEID | DEPARTMENT | RATE |
|---|---|---|---|---|
| ALLEN | PHYLLIS | 4 | ENGINEERING | 8.62 |
| ALLEN | PHYLLIS | 4 | TOOL DESIGN | 8.62 |

## SUBMITTED QUERY

```
 SELECT
       c.LastName , c.FirstName , emain.EmployeeID ,  d.name as Department, eph.Rate
FROM   Department d
       INNER JOIN EmployeeDepartmentHistory edhmain
         ON d.DepartmentID = edhmain.DepartmentID
       INNER JOIN EmployeePayHistory eph
         ON eph.EmployeeID = edhmain.EmployeeID
       INNER JOIN Employee emain
         ON edhmain.EmployeeID = emain.EmployeeID
       INNER JOIN Contact c
         ON emain.ContactID = c.ContactID
WHERE c.FirstName like 'P%'
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| To return Employee History details | To return Employee History details | True | True |

## RETRIEVE RATE COUNT DETAILS

The EmployeePayHistory will have the rate history for each year. Write a query to fetch the number of Rate change  happened for the years 1996, 1997 and 1998. Also display the number of unique rate for the mentioned years.

**Design Rules :**

1. Display Year using column RateChangeDate in YYYY format for the years 1996, 1997 and 1998.

2. The output will have Year, No of Rates Changed and  Distinct Rates.

**Tables:**

EmployeePayHistory

**Sample Output:**

| YEAR | NO OF RATES CHANGED | DISTINCTRATES |
|---|---|---|
| 1997 | 5 | 4 |
| 1998 | 4 | 4 |

## SUBMITTED QUERY

```
 SELECT Year(RateChangeDate) AS [Year],
COUNT([Rate]) AS [No of Rates Changed],
COUNT(DISTINCT Rate) AS [DistinctRates]
FROM [EmployeePayHistory]
WHERE Year(RateChangeDate) in ('1996','1997','1998')
GROUP BY Year(RateChangeDate);
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Retrieve Rate count details | Retrieve Rate count details | True | True |

## TO RETURN EMPLOYEE DETAILS WHOSE STATE IS AB

Write a stored procedure  to select the details of employees like LastName, FirstName, City, StateProvinceCode

**Design Rules :**

1. Use Joins

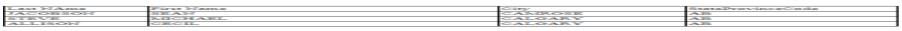
PROCEDURE NAME: GetEmployeeDetails

INPUT PARAMETERS: @StateProvinceCode

**Tables:**

Contact, Employee, EmployeeAddress, Address, StateProvince

Sample Output:



## SUBMITTED QUERY

```
 ALTER PROCEDURE [GetEmployeeDetails]
@StateProvinceCode nvarchar(3)
AS
SELECT c.LastName, c.FirstName, a.City, s.StateProvinceCode
FROM Contact c JOIN Employee e
      ON c.ContactID = e.ContactID
INNER JOIN EmployeeAddress ea
      ON e.EmployeeID = ea.EmployeeID
INNER JOIN Address a
     ON ea.AddressID = a.AddressID
INNER JOIN StateProvince s
      ON a.StateProvinceID = s.StateProvinceID
WHERE s.StateProvinceCode = @StateProvinceCode;
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| To return Employee details whose State is AB | To return Employee details whose State is AB | True | |

## TO RETURN EMPLOYEE DETAILS BASED ON CITY.

Write a query to return the details  Employee details like Name, Department, Manager, City, StateProvinceCode, CountryRegionCode

**Design Rules :**

1. Retreive output where City is ‘Phoenix’

2. Fetch StateprovinceCode, CountryRegionCode from StateProvince table.

3. Fetch Name from Contact table.

4. For Name and Manager columns, use ‘FirstName’ and 'LastName'  seperated by a single space delimiter. Order should be ‘FirstName’  followed by a space, then the ‘LastName’. (Ex: FirstName LastName).

5. If Manager for a particular employee is null then display as ‘CEO’

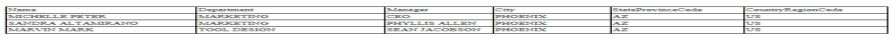
6. Rename the column ‘Name’ from Department table as ‘Department’

7. Order the output by Department.

**Tables:**

Employee, Contact, EmployeeAddress, Address, StateProvince, EmployeeDepartmentHistory, Department

Sample Output:



| Name | Department | Manager | City | StateProvinceCode | CountryRegionCode |
|---|---|---|---|---|---|
| MICHELLE PETER | MARKETING | CEO | PHOENIX | AZ | US |
| SANDRA ALTAMIRANO | MARKETING | PHYLLIS ALLEN | PHOENIX | AZ | US |
| MARVIN MARK | TOOL DESIGN | SEAN JACOBSON | PHOENIX | AZ | US |

## SUBMITTED QUERY

```
 SELECT
  [Name]= c.FirstName +' '+ c.LastName
, [Department] = d.Name
, [Manager]= ISNULL(pc.FirstName+' '+ pc.LastName, 'CEO')
, a.City
, sp.StateProvinceCode
, sp.CountryRegionCode
FROM
            Employee e
  LEFT JOIN Employee emp ON e.ManagerID = emp.EmployeeID
  LEFT JOIN Contact pc ON pc.ContactID = emp.ContactID
  INNER JOIN Contact c ON e.ContactID = c.ContactID
  INNER JOIN EmployeeAddress ea ON e.EmployeeID = ea.EmployeeID
  INNER JOIN Address a ON ea.AddressID = a.AddressID
  INNER JOIN StateProvince sp ON a.StateProvinceID = sp.StateProvinceID
  INNER JOIN EmployeeDepartmentHistory edh
             ON e.EmployeeID = edh.EmployeeID
  INNER JOIN Department d ON edh.DepartmentID = d.DepartmentID
WHERE a.City = 'Phoenix'
ORDER BY  d.Name
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| To return Employee details based on City. | To return Employee details based on City. | True | |

## USING PAGINATION CONCEPT

Write a query to return Employeeid,Managerid,Title,Gender using Pagination concept

**Design Rules :**

1. Ignore the first 5 rows and select the next 3  rows using pagination concept

2. Order the resultset by EmployeeID

**Tables:**

Employee

**Sample Output:**

## SUBMITTED QUERY

```
 select Employeeid,managerid,title,gender from
Employee
order by employeeid
offset 5 rows
fetch next 3 rows only
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Using Pagination concept | Using Pagination concept | True | |

## USER FUNCTIONS

Write a function to select EmployeeName, Gender, EmployeeID, ManagerID, Title

**Design Rules :**

1. For EmployeeName use ‘FirstName’ and 'LastName'  seperated by a single space delimiter. Order

should be ‘FirstName’  followed by a space, then the ‘LastName’. (Ex: FirstName LastName)

FUNCTION NAME: CASQL_Fn_EmployeeDetails

INPUT PARAMETERS: @EmployeeID

**Tables:**

Employee, Contact

**Sample Output:**



## SUBMITTED QUERY

```
 ALTER FUNCTION CASQL_Fn_EmployeeDetails
(@EmployeeID  int)
RETURNS TABLE
AS
RETURN
SELECT FirstName + ' ' + LastName AS 'EmployeeName',
                     E.Gender,
                     EmployeeID,
                     ManagerID ,
                     E.Title
             FROM   Employee AS E
                    JOIN Contact AS C
                      ON C.ContactID = E.ContactID
              WHERE  EmployeeID = @EmployeeID
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| User Functions | User Functions | True | |

## USING RANKING STATEMENTS

Rate incrementation for each ModifiedYear needs to be captured. Write a query to select ModifiedYear,
 Rate and RateRanking (Highest rate to be ranked 1 and so on.)

**Design Rules :**

1. Get Modified Year from the column ModifiedDate.

2. For RateRanking use ranking statements to Rank the Rates (Highest Rate to be ranked 1 and so on) for each Modified Year.

3. Modified Year greater than 2000 only.

4. Order the output by ModifiedDate

5. Refer the sample output for your reference.

**Tables:**

EmployeePayHistory

**Sample Output:**

| MODIFIEDYEAR | RATE | RATERANKING |
|---|---|---|
| 2002 | 29.8462 | 1 |
| 2002 | 25 | 2 |
| 2002 | 12 | 3 |
| 2004 | 29.8462 | 1 |
| 2004 | 13.4615 | 2 |
| 2004 | 12.699 | 3 |

## SUBMITTED QUERY

```
 SELECT
 Year(ModifiedDate) as ModifiedYear,
    [Rate],
 RANK() over(partition by Year(ModifiedDate) order by Rate desc) as RateRanking

FROM EmployeePayHistory where Year(ModifiedDate) >'2000'
order by Year(ModifiedDate)
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Using Ranking Statements | Using Ranking Statements | True | True |

## USING FORMAT FUNCTION

Write a query to return ISO Formatted Date, US Currency, UK Currency.

**Design Rules :**

1. Retreive ModifiedDate as ISO Formatted Date in yyyy-MM-DD format.

2. Select Rate in US & UK currency format and rename the columns as US Currency and UK Currency respectively.  Refer sample output

3. Retreive the output for ModifiedDate  2004

**Tables:**

EmployeePayHistory

**Sample Output:**

| ISO Formatted Date | US Currency | UK Currency |
|---|---|---|
| 2004-07-31 | $12.70 | £12.70 |
| 2004-07-31 | $13.46 | £13.46 |
| 2004-07-31 | $29.85 | £29.85 |

## SUBMITTED QUERY

```
 SELECT
 Year(ModifiedDate) as ModifiedYear,
    [Rate],
 RANK() over(partition by Year(ModifiedDate) order by Rate desc) as RateRanking

FROM EmployeePayHistory where Year(ModifiedDate) >'2000'
order by Year(ModifiedDate)
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Using Format function | Using Format function | False | False |

## USING MERGE STATEMENT

Write a query using MERGE statement and perform delete and update operations in a single query.

**Design Rules :**

1. Relationship between CountryRegion and StateProvince is based on CountryRegionCode.

2. Target table for Update and Delete operation is StateProvince

3.  Perform delete operation in  the merge statement when CountryRegionCode is “ZZ”.

4. Perform update action in the merge statement when CountryRegionCode is “ZY” , update column Name as 'MERGE'.

5. Please follow the order of operation as mentioned above. First delete, then update.

**Tables:**

CountryRegion, StateProvince

**Sample Output:**



## SUBMITTED QUERY

```
 MERGE StateProvince AS statep
USING (SELECT CountryRegionCode FROM StateProvince) AS country
ON country.CountryRegionCode = statep.CountryRegionCode
WHEN MATCHED AND statep.CountryRegionCode = 'ZZ' THEN DELETE
WHEN MATCHED AND statep.CountryRegionCode = 'ZY' THEN UPDATE SET statep.Name = 'MERGE';
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Using Merge statement | Using Merge statement | True | |

## USING PIVOT FUNCTION

Write a query to select the total sum of Rate for each employee for the given modifed year using PIVOT concept. Select the Employeeid, Rate, ModifiedYear from EmployeePayHistory table.

**Design Rules :**

1. For employeeid less than and equal to 4.

2. Get ModifiedYear from the column ModifiedDate.

3. For ModifiedYear  [2004],[2002],[1997] only

**Tables:**

EmployeePayHistory

**TempTableName:**

#PayHistoryPivotResult

Sample Output:

## SUBMITTED QUERY

```
 SELECT *
      INTO #PayHistoryPivotResult
FROM
  (SELECT EmployeeID,Rate,
    YEAR(ModifiedDate) AS ModifiedYear
  FROM EmployeePayHistory
  where employeeid<=4) AS PayHistory
  PIVOT(SUM(Rate)
  FOR ModifiedYear IN([2004],[2002],[1997])
  ) AS PivotPayHistory;

  select * from #PayHistoryPivotResult
```

## TEST CASE EXECUTION

| Test Case Name | Test Case Description | Primary TestCase | Secondary TestCase |
|---|---|---|---|
| Using Pivot function | Using Pivot function | True | True |