**Basic Databases – Report05**

Wroclaw University of Technology, Date: December 7, 2018

|  |  |  |
| --- | --- | --- |
| Student: | Email:241018@student.pwr.edu.pl | Grade |
| Identifier | 241018 | ? |
| First name | Siamion |
| Last name | Rondzel |

The tasks contain exercises to practice two subjects:

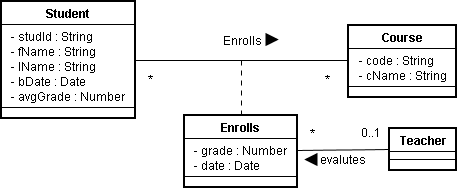
1. Conceptual data model and its implementation using SQL
2. Retrieving Information using VIEWs
3. Set operations
4. The OVER clause
5. Pivoting data

**Additional note:**

**Please carefully read through the tasks and decide which tasks you want to solve with my help**

**---------------------------------------------------------------------------------------------------------------------------- ----------------------**

# Problem 1



1. Complete definition of classes (add needed properties)



1. Implement the data model in the MS SQL 20017 server - transform classes and relationships from the conceptual data model into the set of tables including appropriate integrity constraints (primary key, alternate key, foreign key, and check)

CREATE TABLE Student(

studId CHAR(7) NOT NULL,

fName CHAR(7) NOT NULL,

lName CHAR(7) NOT NULL,

bDate DATE,

avgGrade DECIMAL(2,1),

phoneNumber CHAR(11),

emailAddress VARCHAR(32),

CONSTRAINT pk\_Student PRIMARY KEY (studId)

);

CREATE TABLE Course(

code CHAR(5) NOT NULL,

cName VARCHAR(32) NOT NULL,

creditHours INT NOT NULL,

creditPoints INT NOT NULL,

CONSTRAINT pk\_Course PRIMARY KEY (code)

);

CREATE TABLE Enrolls(

id INT NOT NULL IDENTITY(1,1),

grade DECIMAL(2,1),

date DATE NOT NULL,

studId CHAR(7) NOT NULL,

code CHAR(5) NOT NULL,

CONSTRAINT fk1\_Enrolls FOREIGN KEY (studId) REFERENCES Student(studId),

CONSTRAINT fk2\_Enrolls FOREIGN KEY (code) REFERENCES Course(code),

CONSTRAINT pk\_Enrolls PRIMARY KEY (id)

);

CREATE TABLE Teacher(

teachId CHAR(7) NOT NULL,

fName CHAR(7) NOT NULL,

lName CHAR(7) NOT NULL,

bDate DATE,

phoneNumber CHAR(11) NOT NULL,

emailAddress VARCHAR(32) NOT NULL,

dateOfEmployement DATE NOT NULL,

CONSTRAINT pk\_Teacher PRIMARY KEY (teachId)

);

CREATE TABLE Evaluates(

id INT NOT NULL IDENTITY(1,1),

teachId CHAR(7) NOT NULL,

id\_enrollment INT NOT NULL,

CONSTRAINT fk1\_Evaluates FOREIGN KEY (teachId) REFERENCES Teacher(teachId),

CONSTRAINT fk2\_Evaluates FOREIGN KEY (id\_enrollment) REFERENCES Enrolls(id),

CONSTRAINT pk\_Evaluates PRIMARY KEY (id)

);

1. Insert some records into each table

INSERT INTO Student VALUES

('2098174','Karl','Jeims','1998-10-3',3.8,'48123456789','email@mail.com'),

('2098175','Ian','Kupel','1997-11-13',4.8,'48121111789','email2@mail.com'),

('2098176','Krzystyna','Jon','1999-10-3',4.0,'48007456789','email3@mail.com')

;

INSERT INTO Course VALUES

('20981','Course#1',30,2),

('20982','Course#2',45,5),

('20983','Course#3',60,6)

;

INSERT INTO Enrolls VALUES

(3.7,'2017-9-27','2098174','20982'),

(4.7,'2016-9-27','2098175','20981'),

(4.0,'2017-9-28','2098174','20981'),

(3.0,'2017-9-27','2098176','20981')

;

INSERT INTO Teacher VALUES

('9998160','John','Kladt','1968-10-3','48123456123','email@mail.com','1999-1-10'),

('9998161','Pawel','Jun','1987-11-13','48121111321','email2@mail.com','2013-10-8'),

('9998162','Alena','P.D.','1990-10-3','48076556789','email3@mail.com','2018-10-10')

;

INSERT INTO Evaluates VALUES

('9998161',2),

('9998162',3)

;

SELECT \* FROM Student;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| studId | fName | lName | bDate | avgGrade | phoneNumber | emailAddress |
| 2098174 | Karl | Jeims | 1998-10-03 | 3.8 | 48123456789 | email@mail.com |
| 2098175 | Ian | Kupel | 1997-11-13 | 4.8 | 48121111789 | email2@mail.com |
| 2098176 | Krzystyna | Jon | 1999-10-03 | 4.0 | 48007456789 | email3@mail.com |

SELECT \* FROM Course;

|  |  |  |  |
| --- | --- | --- | --- |
| code | cName | creditHours | creditPoints |
| 20981 | Course#1 | 30 | 2 |
| 20982 | Course#2 | 45 | 5 |
| 20983 | Course#3 | 60 | 6 |

SELECT \* FROM Enrolls;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | grade | date | studId | code |
| 1 | 3.7 | 2017-09-27 | 2098174 | 20982 |
| 2 | 4.7 | 2016-09-27 | 2098175 | 20981 |
| 3 | 4.0 | 2017-09-28 | 2098174 | 20981 |
| 4 | 3.0 | 2017-09-27 | 2098176 | 20981 |

SELECT \* FROM Teacher;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| teachId | fName | lName | bDate | phoneNumber | emailAddress | dateOfEmployement |
| 9998160 | John | Kladt | 1968-10-03 | 48123456123 | email@mail.com | 1999-01-10 |
| 9998161 | Pawel | Jun | 1987-11-13 | 48121111321 | email2@mail.com | 2013-10-08 |
| 9998162 | Alena | P.D. | 1990-10-03 | 48076556789 | email3@mail.com | 2018-10-10 |

SELECT \* FROM Evaluates;

|  |  |  |
| --- | --- | --- |
| id | teachId | id\_enrollment |
| 1 | 9998161 | 2 |
| 2 | 9998162 | 3 |

--select the student's name, course name, grade, name of the teacher who evaluates this course, his phone and email address, for the courses, that are evaluated by a teacher

SELECT S.fName + ', ' + S.lName AS 'Student Name', cName AS 'Enrolled Course', grade AS Grade, T.fName + ', ' + T.lName AS 'Teacher Name',

T.phoneNumber AS "Teacher's phone", T.emailAddress AS "Teacher's email"

FROM Student AS S, Teacher AS T, Course AS C, Enrolls AS E, Evaluates AS Ev

WHERE S.studId=E.studId AND E.code=C.code AND E.id=Ev.id\_enrollment AND Ev.teachId=T.teachId;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student Name | Enrolled Course | Grade | Teacher Name | Teacher's phone | Teacher's email |
| Ian, Kupel | Course#1 | 4.7 | Pawel, Jun | 48121111321 | email2@mail.com |
| Karl, Jeims | Course#1 | 4.0 | Alena, P.D. | 48076556789 | email3@mail.com |

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

Source data**: Database: AdventureWorks2017**

# Task 1

Table: SalesOrderHeader, SalesOrderDetail

Create a view that allows retrieving information from the database of the total quantity orders in the context of employee and year

# Solution

CREATE VIEW task2\_view AS

SELECT H.SalesPersonID, YEAR(OrderDate) AS Year, SUM(OrderQty) AS NoOfOrders

FROM Sales.SalesOrderHeader AS H, Sales.SalesOrderDetail AS D

WHERE SalesPersonID IS NOT NULL AND H.SalesOrderID=D.SalesOrderID

GROUP BY SalesPersonID, YEAR(OrderDate);

|  |  |  |
| --- | --- | --- |
| **SalesPersonID** | **Year** | **NoOfOrders** |
| **281** | **2014** | **2050** |
| **279** | **2011** | **1681** |
| **285** | **2013** | **752** |

Rec:3/58

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

# Task 2

Table: **?**

Write a query that returns customer and employee pairs that had order activity (number of orders > 0) in January 2012 but not in May 2014

# Solution

SELECT CustomerID, SalesPersonID

FROM Sales.SalesOrderHeader AS H

WHERE MONTH(OrderDate)=1 AND YEAR(OrderDate)=2012 AND SalesPersonID IS NOT NULL

AND (SalesPersonID NOT IN(

SELECT SalesPersonID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=5 AND YEAR(OrderDate)=2014 AND CustomerID = H.CustomerID AND SalesPersonID IS NOT NULL

)

AND CustomerID NOT IN(

SELECT CustomerID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=5 AND YEAR(OrderDate)=2014 AND SalesPersonID = H.SalesPersonID AND SalesPersonID IS NOT NULL

)

)

--OR USING VIEWS

CREATE VIEW task3\_1\_view AS

SELECT CustomerID, SalesPersonID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=1 AND YEAR(OrderDate)=2012 AND SalesPersonID IS NOT NULL

CREATE VIEW task3\_2\_view AS

SELECT CustomerID, SalesPersonID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=5 AND YEAR(OrderDate)=2014 AND SalesPersonID IS NOT NULL

SELECT CustomerID, SalesPersonID

FROM task3\_1\_view AS H

WHERE (SalesPersonID NOT IN(

SELECT SalesPersonID

FROM task3\_2\_view

WHERE CustomerID = H.CustomerID

)

AND CustomerID NOT IN(

SELECT CustomerID

FROM task3\_2\_view

WHERE SalesPersonID = H.SalesPersonID

)

)

|  |  |
| --- | --- |
| **CustomerID** | **SalesPersonID** |
| **29769** | **277** |
| **29523** | **277** |
| **30100** | **276** |

Rec:3/128

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

# Task 3

Write a query that returns customer and employee pairs that had order activity in both January 2014 and May 2014 but not in 2013

# Solution

SELECT CustomerID, SalesPersonID

FROM Sales.SalesOrderHeader AS H

WHERE MONTH(OrderDate)=1 AND YEAR(OrderDate)=2014 AND SalesPersonID IS NOT NULL

AND (SalesPersonID NOT IN(

SELECT SalesPersonID

FROM Sales.SalesOrderHeader

WHERE YEAR(OrderDate)=2013 AND CustomerID = H.CustomerID AND SalesPersonID IS NOT NULL

)

AND CustomerID NOT IN(

SELECT CustomerID

FROM Sales.SalesOrderHeader

WHERE YEAR(OrderDate)=2013 AND SalesPersonID = H.SalesPersonID AND SalesPersonID IS NOT NULL

)

)

AND (SalesPersonID IN(

SELECT SalesPersonID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=5 AND YEAR(OrderDate)=2014 AND CustomerID = H.CustomerID AND SalesPersonID IS NOT NULL

)

AND CustomerID IN(

SELECT CustomerID

FROM Sales.SalesOrderHeader

WHERE MONTH(OrderDate)=5 AND YEAR(OrderDate)=2014 AND SalesPersonID = H.SalesPersonID AND SalesPersonID IS NOT NULL

)

)

# Rec:0/0

------------------------------------------------------------------------------------------------------------------- ------------------------------------------------

# Task 4

Write a query that returns data according to the following definition:

custID, Name (Last Name, First name), Gender, “2012. 04 –2013. 04” (Difference of orders in April 2012 and April 2013)

# Solution

CREATE VIEW April2012Orders AS

SELECT CustomerID, COUNT(\*) AS April2012Orders

FROM Sales.SalesOrderHeader

WHERE YEAR(OrderDate)=2012 AND MONTH(OrderDate)=4

GROUP BY CustomerID

CREATE VIEW April2013Orders AS

SELECT CustomerID, COUNT(\*) AS April2013Orders

FROM Sales.SalesOrderHeader

WHERE YEAR(OrderDate)=2013 AND MONTH(OrderDate)=4

GROUP BY CustomerID

SELECT T1.CustomerID, P.LastName + ', ' + P.FirstName AS 'Name (Last Name, First name)',

COALESCE(T1.April2012Orders-T2.April2013Orders, T1.April2012Orders) AS '“2012. 04 –2013. 04” (Difference of orders in April 2012 and April 2013)'

FROM April2012Orders AS T1

LEFT JOIN April2013Orders AS T2 ON T1.CustomerID=T2.CustomerID

JOIN Sales.Customer AS C ON C.CustomerID = T1.CustomerID

JOIN Person.Person AS P ON P.BusinessEntityID = C.PersonID;

|  |  |  |
| --- | --- | --- |
| **CustomerID** | **Name (Last Name, First name)** | **“2012. 04 –2013. 04” (Difference of orders in April 2012 and April 2013)** |
| **11395** | **Gutierrez, Beth** | **1** |
| **11412** | **Bryant, Sydney** | **1** |
| **11421** | **Sun, Amy** | **1** |

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

# Task 5

Write a query that returns the total sum of orders for each customer according to the following definition:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CustID | Name | No | Total due | Date | Sum of total due |
| 11000 | Yang, Jon | 1 | 3756,989 | 2011-07-03 | 9115.13 |
| … | … | 2 | … | … | … |
| 11001 | Huang, Eugene | 1 | 2674,0227 | 2013-06-30 | 7054.19 |
| … |  | 2 |  |  | 7054.19 |
| 11002 | Torres, Ruben | 1 | 3756,989 | 2011-06-21 | 8966.01 |
| … | … | … | … | … | … |
| 11003 | Zhu, Christy | 1 | 2674,4757 | 2013-10-22 | 8993.92 |
| … | … | … | … | … | … |

------------------------------------------- Explanation / Example ----------------------------------------------

The OVER clause exposes a window of rows to certain kinds of calculations. Aggregate and ranking functions are the types of calculations that support the OVER clause. In this case you don’t have to group the data (GROUP BY)

## SELECT …

**, <aggregate / ranking function> (aggregation element>) OVER([PARTITION BY <list of attributes>**

**[ORDER BY <list of attributes>]]) AS <column alias>**

**, …**

**FROM <source data>**

Example:

SELECT so.SalesPersonID, p.LastName +', ' + p.FirstName Name

, ROW\_NUMBER() OVER(PARTITION BY so.SalesPersonID ORDER BY so.SalesPersonID) No

, so.SubTotal, CAST(so.DueDate AS DATE) Date

, COUNT(\*) OVER(PARTITION BY so.SalesPersonID) "No of records" FROM [Sales].[SalesOrderHeader] so JOIN [Person].[Person] p

ON so.[SalesPersonID] = p.[BusinessEntityID] ORDER BY so.SalesPersonID, No;

Exemplary result of the query:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SalesPersonID | Name | No | SubTotal | Date | No of records |
| 274 | Jiang, Stephen | 1 | 20544,7015 | 2011-07-13 | 48 |
| 274 | Jiang, Stephen | 2 | 2039,994 | 2011-08-13 | 48 |
| 274 | Jiang, Stephen | 3 | 4194,589 | 2011-10-13 | 48 |
| 274 | Jiang, Stephen | 4 | 2146,962 | 2011-10-13 | 48 |
| … | … | … | … | … | … |
| 275 | Blythe, Michael | 1 | 2942,418 | 2014-05-13 | 450 |
| 275 | Blythe, Michael | 2 | 5496,018 | 2014-05-13 | 450 |
| 275 | Blythe, Michael | 3 | 2995,188 | 2014-05-13 | 450 |
| 275 | Blythe, Michael | 4 | 3595,188 | 2014-05-13 | 450 |
| … | … | … | … | … | … |

-----------------------------------------------------------------------------------------------------

# Solution

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

# Task 6

Write a query that returns the count of orders for each employee who prepared orders in 2012, 2013, and 2014

------------------------------------------- Explanation / Example ----------------------------------------------

Pivoting means rotating data from a state of rows to a state of columns. Every pivoting request involves three logical processing phases:

* 1. A grouping phase
  2. A spreading phase
  3. An aggregating phase

The general form of a query with PIVOT operator is:

## SELECT …

**FROM <source data> PIVOT**

**(**

**<Aggregate function> (aggregation element>)**

**FOR <spreading element> IN (<list of target columns>)**

**) AS <result table alias>**

Exemplary result of the query

* Simplified solution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| empId | 2011 | 2012 | 2013 | 2014 |
| 284 | 0 | 24 | 82 | 34 |
| 278 | 30 | 80 | 89 | 35 |
| 281 | 33 | 74 | 98 | 37 |
| 275 | 65 | 148 | 175 | 62 |

* Extended solution (more readable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employer | empId | 2011 | 2012 | 2013 | 2014 |
| Mensa-Annan, Tete | 284 | 0 | 24 | 82 | 34 |
| Vargas, Garrett | 278 | 30 | 80 | 89 | 35 |
| Ito, Shu | 281 | 33 | 74 | 98 | 37 |
| Blythe, Michael | 275 | 65 | 148 | 175 | 62 |
| Mitchell, Linda | 276 | 46 | 151 | 162 | 59 |

-------------------------------------------------------------------------------------------------------------

# Solution

----------------------------------------------------------------------------------------------------------------------------- --------------------------------------

# Task 7

Define query using SQL table value constructor with 5 records as elements of a dictionary. Each record should contain a pair of values: **id** (from 1 through 5) and a **name** of programming language, e.g. (1, 'SQL'), (2, 'Python'), …

|  |  |
| --- | --- |
| id | language |
| 1 | SQL |
| 2 | Python |
| … | … |

------------------------------------------- Syntax ----------------------------------------------

## SELECT \* FROM

**(VALUES**

**(…)**

**) AS <name of table> (<definition of table structure>)**

**Solution**