|  |  |
| --- | --- |
| **Project** | Description: LogoBINUS-University |
| DBH3 |
| **Periode Berlaku** Semester Genap 2020/2021  ***Valid on*** *Even Semester Year 2020/2021* | **Software Laboratory Center**  **Assistant Recruitment 21-2** |

## Materi

*Material*

* Join
* Subquery
* Alias Subquery
* Union
* Insert
* Delete
* Stored Procedure
* Function
* Cursor
* Trigger

## Soal

*Case*

**JPKeyboard**

JPKeyboard is a local keyboard store that just recently opened. JPKeyboard has just implemented a database to store their daily transactions. As the first Database Administrator in JPKeyboard, you are assigned to finish 40 tasks using Microsoft SQL Server as the database management system. The task are based on their Entity Relationship Diagram (ERD):

Diagram

Description automatically generated

Figure 1. Entity Relationship Diagram

1. Display ProductName and TotalPurchase classes for each product that sells more than average.

**(Alias Subquery, Avg, Count, Join)**

Table

Description automatically generated

1. Show **ProductNo**(Obtained by **adding** **‘product’** and **productid**), **ProductName** and **ProductDescription,** and **TotalQtyBought**(Obtained by totalling qty from every transaction) for every product bought by **Clarissa Chuardi**.

**(Sum, Join, Cast)**

Graphical user interface, text, application

Description automatically generated

1. Display **all customers** that never bought **“Kailh Box Dark yellow”**, that have ever bought **“Keychron K8”**.

**(Exists, Not Exist, Subquery, Join)**



1. Display **CustomerName**, **CustomerDOB(Obtained from customerDOB in DD MMM YYYY Format)** and **CustomerGender(Obtained from CustomerGender’s first letter)** for every customer that **ever bought** a **product** that **contains gateron** in its **productname** and **never** **bought** a **product** that **contains space65** in its **productname**.

**(Convert, Exist, Not Exist,Subquery, Left)**

**Table

Description automatically generated with medium confidence**

1. Show **all** **tehnicianname**, and every **distinct** **service** **the technician has ever done**.

**(XML, Stuff, Alias Subquery, Join)**

Graphical user interface, text, application

Description automatically generated

**(54 rows)**

1. Show **ProductTypeName** **for all** **ProductType** and **all** of **it’s product**.

**(XML, Stuff, Alias Subquery)**

Graphical user interface, text

Description automatically generated

1. Show CustomerName for all Customers and all products the customer bought.

**(XML, Stuff, Alias Subquery, Join)**

Graphical user interface, text, application

Description automatically generated

1. Show **StaffId**, **StaffName**, **TotalTransaction** and **Total Revenue** (obtained by counting how much revenue each staff made) for **every** **staff** who **made more than 1 transaction this year**, **ordered by TotalRevenue**.

**(Sum, Count, Alias Subquery, Order By, Join)**

Table

Description automatically generated

1. Display Staff's **Nickname** (**obtained** from the first 2 characters of StaffName and staff id), and **MiddleName** (obtained by taking character after the first space until character before second space in StaffName) for every staff whose name contains at least 3 words and hasn’t served any female customer.

**(Left, Cast, Substring, Charindex, Len, Not Exists, In, Not Like, Like)**

Table

Description automatically generated

1. Display **CustomerId**, **LastName** and **TotalRevenue**(calculated from total money spent by the customer on products and services) for **all customers**.

**(Reverse, Substring, Charindex, Alias Subquery, Sum, Left Join)**

**Table

Description automatically generated**

1. Show **technicianName** (obtained from technician's **first name**), **TechnicianRoundedAge** (Obtained from year **difference between technician dob** and **the date now**)

Notes:

* If the technician only have 1 word in his/her name, make sure it still appears
* Make sure the Age is real (For example, if someone is born at December 2001 that means his/her age is still 19)

**(Case, Isnull, Nullif, Charindex, len, Datediff, Day, Month, Year, Getdate, Cast)**

Table

Description automatically generated

(25 Rows)

1. Show:

* **CustomerWhoeNeverMadeServiceTransaction**(Obtained by finding customer who have 0 servicetransaction)
* **TechnicianWhoNeverMadeServiceTransaction**(Obtained by finding technician who have 0 servicetransaction)**,**
* **CustomerWhoNeverMadeProductTransaction** (Obtained by displaying customer who have 0 producttransaction)
* **StaffWhoNeverMadeProductTransaction**(Obtained by displaying staff who have 0 product transaction).

**(Left Join, Count, Alias Subquery)**



1. Show every **customer name** and **total address** **the** **customer owns** in a **pivot table**.

**(Pivot, Count, Join)**

Graphical user interface, table

Description automatically generated

1. Display all **service name** and it’s **revenue** in a **pivot table**.

**(Pivot, Sum, Join)**



1. Display all (Service and Product) transactions’ revenue at **october, november and december** at **2020**.

**(Pivot, DateName, Month, Year, Sum, Join)**

****

1. Display CustomerName, TotalProductTransaction(obtained from counting total customer transaction), TotalMoneySpentOnProduct(obtained from counting all the money customers spent on product) for every producttransaction that happened in 2020. Then convert the output into xml format.

**(Count, Sum, Year, For XML, Join)**

Table Output:

A picture containing graphical user interface

Description automatically generated

Final Output (XML)



1. Open keyboard.json, then select the data into a table format.

**(json\_value, json\_query)**



1. Create a temp table named **ArrivedKeyboardStock**

|  |  |
| --- | --- |
| **FieldName** | **DATATYPE** |
| ProductName | VARCHAR(80) |
| ProductPrice | INT |
| ProductStock | INT |
| ProductWeight | FLOAT |
| ProductDescription | VARCHAR(300) |

Then take the data from **keyboard.csv** and insert it into **ArrivedKeyboardStock**

After doing so, select from **ArrivedKeyboardStock** where ProductDescription doesn’t contain the word **‘membrane’**.

**(Bulk Insert)**

A picture containing table

Description automatically generated

1. Display **MiddleName**(Obtained from **Customer middle name**), **Initial**(Obtained from **first 2 characters** from **Customer Email**) for every **customer** who bought **more than 3 products** and **more than 3 services**.

**(Intersect, Right, Left, Charindex, Upper, Len, Join)**

Table

Description automatically generated

1. Display **Initial**(Obtained from **first character** in the **first name** and **first character** in the **second name**), and **Transaction Date** for every Customer who made **Product Transaction** and **Service Transaction** on the same date.

**(Left, Substring, Charindex, Convert, Intersect, Join)**

Application

Description automatically generated with low confidence

1. **Display** **Title** which will **contain** **Maximum Product Price, Maximum Service Base Price, Minimum Product Price and Minimum Service Base Price** and **display** **Value** which will show the value for each **title**.

**(Union, Max, Min, Join)**

Text, table

Description automatically generated with medium confidence

1. Display **Id**, **Name** and **TransactionCount** for top **3 staff** who made the **most producttransaction** **this year** and top **3 technician** who made the **most servicetransaction** **this year**.

**(Count, Year, Getdate, Union, Alias Subquery, Join)**

Graphical user interface, text, application

Description automatically generated

1. Display **TechnicianName** for every **technician** who ever made a transaction with **Johanes Peter Vincentius** and never made any transaction with **Brandon Julio Thenaro**.

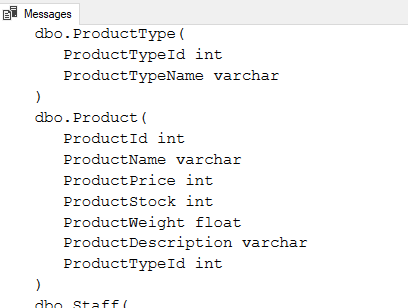
**(Join, Year, Except)**

Table

Description automatically generated with medium confidence

1. Create a cursor to show all tables, all attributes and datatypes inside the table.

**(Cursor, While)**



1. Create cursor to retrieve **First Customer**, **Last Customer** and **Mid Customers**(Obtained from fetching the once every **eight data** from **absolute** cursor position ) for **servicetransactions** in 2021.

**(Scroll Cursor, While, Join)**

Table

Description automatically generated

1. Create procedure **PrintMasterTableSizes** to get **all** **master table size.**

**(Alias Subquery, Create Procedure)**

EXEC PrintMasterTableSizes



1. Create procedure **ShowServices** to show

**Customer Name**,

**Total money spend**(Obtained from counting servicebaseprice\*qty + ServiceAdditionalPrice),

**Average Money Spend**(Obtained from avg of servicebaseprice\*qty + ServiceAdditionalPrice),

**Customer Most Ordered Service**(Obtained from the most frequently ordered service),

**Customer Last Service Transaction Date**(Obtained from the latest date of transactionService) and **all services** purchased by **a customer.**

**(Create Proc, Sum, Cast, While, Cursor, While)**

EXEC ShowServices 5

Table

Description automatically generated

1. Create procedure **SearchProduct** to search a product by a keyword that receives a parameter **searchString**. The stored procedure will display **Product Id,** **Product Name,** **Product Price,** **Product Availability**(If product **stock is 0**, **Product is currently not available for purchase, please wait for restock**. If product **stock is more than 0**, **Product is available for purchase**) for every product that **ProductName/ProductTypeName/ProductDescription** contains searchString.

**(Create Proc, Like, Join, Cursor, While)**

EXEC SearchProduct ‘rgb’

**Text

Description automatically generated with low confidence**

1. Create Procedure **PrintSquare** to print a square. The procedure will receive **SquareSize** as its parameter, and will then **print a square** based on its parameter.

**(Create Proc, While)**

Chart

Description automatically generated

1. Create a Trigger **ServiceOrderAlert** to show if anyone ordered a service. The trigger will **print** **customer name** for the customer who just made a transaction and **tell** that the customer has made his/her **n-th service order**.

**(Create Trigger)**

BEGIN TRAN

INSERT INTO ServiceTransaction VALUES(5,3,GETDATE())

ROLLBACK

Graphical user interface, text, email

Description automatically generated

1. Create trigger **PreventChange** to **prevent customer** with **id 5** from being **deleted/updated**.

**(Create Trigger)**

BEGIN TRAN

DELETE FROM Customer WHERE CustomerId=5

ROLLBACK

Text

Description automatically generated

1. Create trigger **ProductTrigger** to **notify** if there is an **insert/update/delete** dml on the **product**. The trigger will **determine** if the **dml** is an **insert/update/delete**. After determining the dml, the trigger will then:

**(Create Trigger, Cast, Join)**

* If it is an **insert**, the trigger will show the **inserted data**

BEGIN TRAN

INSERT INTO PRODUCT VALUES('testing', 92,2,212,'ini produk testing doang sih',2)

ROLLBACK

Text

Description automatically generated

* If it is an **update**, the **trigger** will show the **new data** and **the old data**

BEGIN TRAN

UPDATE PRODUCT

SET ProductName='testingupdated',

ProductPrice= 1234123,

ProductStock=432,

ProductWeight=11,

ProductDescription='ini produk testing testing'

WHERE ProductName='testing'

Graphical user interface, text, application

Description automatically generated

* If it is a **delete**, the trigger will show the **deleted data**

BEGIN TRAN

INSERT INTO PRODUCT VALUES('testing', 92,2,212,'ini produk testing doang sih',2)

ROLLBACK

Graphical user interface, text, application, email

Description automatically generated

1. Create trigger **CustomerAddress** to **notify** if a **customer** just added an **address**. The trigger will also show the customer’s total address and the details of the address he/she just inserted.

**(Create Trigger, Count, Cast)**

BEGIN TRAN

INSERT INTO Address VALUES(1,'Rumah','jalan makaliwe','jakarta','indonesia')

ROLLBACK

Text

Description automatically generated

1. Create **Trigger** **TriggerStaffDelete** to implement **softdelete** on **staff** table. The trigger will implement softdelete by **inserting** the **deleted** data into **a table** named **resignedStaffs**. The trigger will **validate** if the table **resignedStaffs** **exist**. **If** it **doesn’t exist**, the trigger will **create** the **table** and **insert** the **deleted** **data** to **resignedStaffs**.

**(Create Trigger, Object\_id, Select into)**

Note: the table **resignedStaffs** **doesn’t exist** before **this trigger is triggered**.

BEGIN TRAN

DELETE FROM Staff WHERE StaffId=3

ROLLBACK

Text

Description automatically generated

SELECT \* FROM resignedStaffs



1. Create **function** **ProductTypeRevenue**, which will receive a **year** as a **parameter**. The function will **return a table** showingthe **revenue coming** from **every product** **according** to the **year parameter**.

**(Create Function, Join, Sum, Cast, Year)**

SELECT \* FROM dbo.ProductTypeRevenue(2021)

Table

Description automatically generated

1. Create **function Factorial** that will receive a **baseNumber** as a parameter. The function will then return an INT, which will be **baseNumber** **factorial**.

**(Create Function)**

SELECT dbo.Factorial(10)



1. Create **table valued function** **SelectRandomItem** that will receive **ProductTypeId** and **RandomInt(RAND() function cannot be called inside a function, so we will receive it as a parameter)** as it’s parameter. **Based** on **ProductTypeId**, the function will then select **one** **random** **product**.

**(Create Function, Row\_number, Cast, Count)**

SELECT \* FROM dbo.SelectRandomItem(1,RAND())



1. Create **scalar** **function** **CalculateTotalPrice** that will receive **itemList** as a parameter. The function will then **calculate** the **total price** of the transaction by counting all **product price \* qty**.

**(Create Function, Sum, Join)**

Notes:

* The parameter itemList will have the data type **ScannedProductsTable** (which is already available from jp\_keyboard database). ScannedProductsTable details:

|  |  |
| --- | --- |
| AttributeName | DataType |
| ProductId | INT |
| qty | INT |

* From the table above, you will be able to **calculate** the **total price** by finding the **productprice** and **multiplying** it with **qty**.

DECLARE @prodTable AS ScannedProductsTable

INSERT INTO @prodTable

VALUES(2,4),(5,10)

SELECT dbo.CalculateTotalPrice(@prodTable)

A picture containing shape

Description automatically generated

1. Create a **table** **valued** **function** called **GetPlayButton** that receives **size** as parameter. The function **returns** a **table** that is filled with rows that resembles a **play** **button** in one column. If the parameter is larger than 255, then the function will only print “Size may not be larger than 255”.

**(create function, while, insert)**

SELECT \* FROM dbo.GetPlayButton(4)

Table

Description automatically generated

SELECT \* FROM dbo.GetPlayButton(300)



1. Create **cursor** to drop **all** **stored procedures** and **all triggers** from the **database**.

**(Cursor, While)**

Text

Description automatically generated with medium confidence