Darren Excel Hartanto Oey - 2501999223

1. The main function of maintaining database security is to protect the data inside database against the unintentional and intentional threat such as hijacker, because Database is a a very valuable asset because it consist of whole bunch of data that most of it are privacy, Imagine if there's a person who try to hijack the database in the hospital to get it's data that consists of all patient personal information with it's sickness and what if that person is a well famous person, it could be troublesome if the hijacker tried to blackmail.

There's many useful action step that could be useful in maintaining the database security :

1. Needs To patch SQL Server Regularly

This is the most needed one because hacker, virus, malware and other dangerous entity are scattered across the internet everyday it's growing much more stronger and deadlier so it needs to keep up to date with it's SQL server software because time deployment of the current SQL service pact, cumulative update and critical security will also highly advance the stability and the security of the database.

2. Needs to regularly to backup the system to the physical storage

The needs to backup the system is a great way to maintain the system because when the data is being backed up in a physical storage, it's data will remain to save and if the threat happened it won't lose all of the data because the data being backed up regularly.

3. Do stress testing on the database

This stress testing is also important and needs to be run once in 3 month to test the database system endurance when there is DdoS attack from hijackers / hacker and to test database overload.

4. Controlling who could access the database

When picking the server authentication that could choose between integrated (windows) or sql build in choose the first option because integrated (windows) option is authentication encrypts messages to validate user while SQL server authentication is sharred across the network and leave them un protected (use integrated windows authentication)

(NEVER USE SHARED USER ACCOUNT) https://www.dbta.com/Editorial/Trends-and-Applications/5-Top-Tips-to-Maintain-the-Security-of-Your-Database-Environment-100526.aspx

Reference :

https://www.dbta.com/Editorial/Trends-and-Applications/5-Top-Tips-to-Maintain-the-Security-of-Your-Database-Environment-100526.aspx

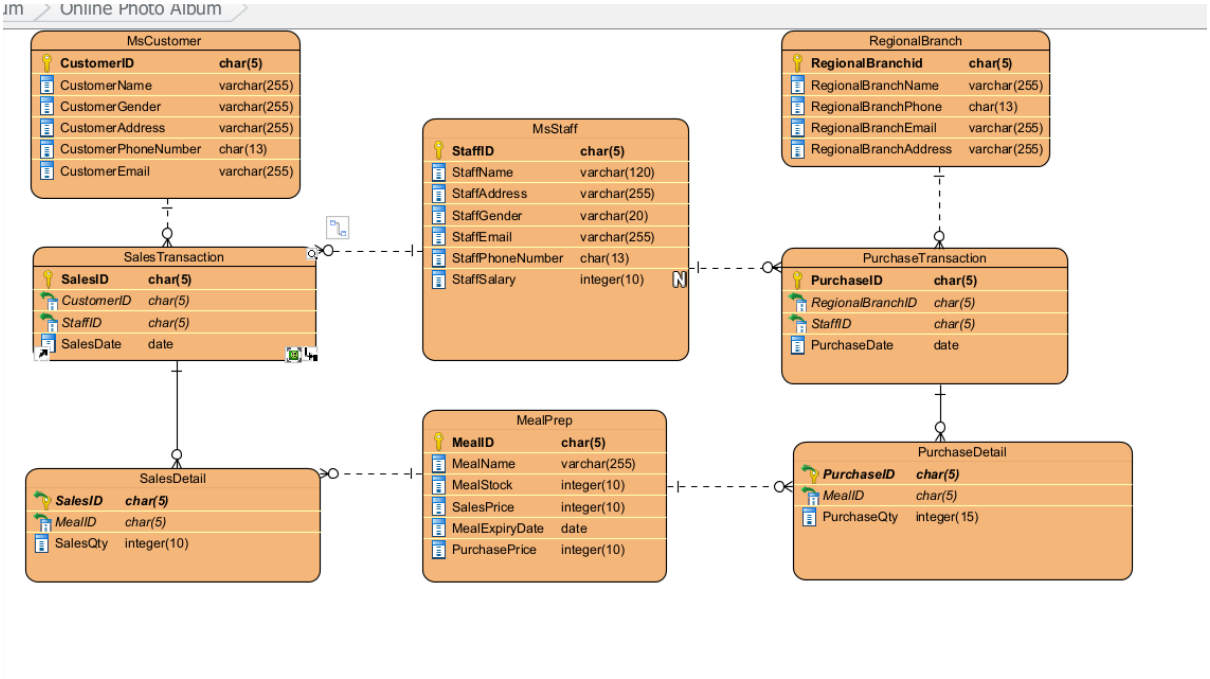
Database Security – School of Information Systems (binus.ac.id)

2. The difference :

Parameter	JOIN	UNION
Basic	If two separate tables or relations share an attribute or field, the JOIN clause **join**s the tuples and their attributes from both.	The UNION clause **combines** the tuples of those relations that you will find in a query.
Type	There are four major types of the JOIN clause- the LEFT, RIGHT, FULL OUTER, and INNER JOIN.	There are two major types of the UNION clause- the UNION and the UNION ALL.
Effect	The length of the tuples that included in the relations is to be longer than the resultant tuple obtained after using the JOIN clause.	The total number of resultant tuples obtained after applying the UNION clause is always more when compared to the tuples present in both the relations of a query.
Application Conditions	can only use the JOIN clause if two relations involved in it at least have one attribute in common.	can use the UNION clause when the total number of columns in a query is the same while the corresponding attribute has a similar domain.

Taking From my Previous database project from LAB

Here's the ERD



JOIN

```sql
--2
SELECT StaffName, [SalesDate] = CONVERT(VARCHAR,SalesDate,106), [Total Sales] = SUM(SalesQty)
FROM MsStaff ms
JOIN SalesTransaction st ON st.StaffID = ms.StaffID
JOIN SalesDetail sd ON sd.SalesID =  st.SalesID

GROUP BY StaffName, SalesDate
```

Here means that I want to take the 'sales Date' inside the Sales Transaction Table to connect it to the Staff name so I need to make a connection from the MsStaff Table and The Sales Transaction Table and there is a common coloumn name called StaffID that connecting The MsStaff and Sales Transaction Table, and so on and so forth here's the result

Results | Messages

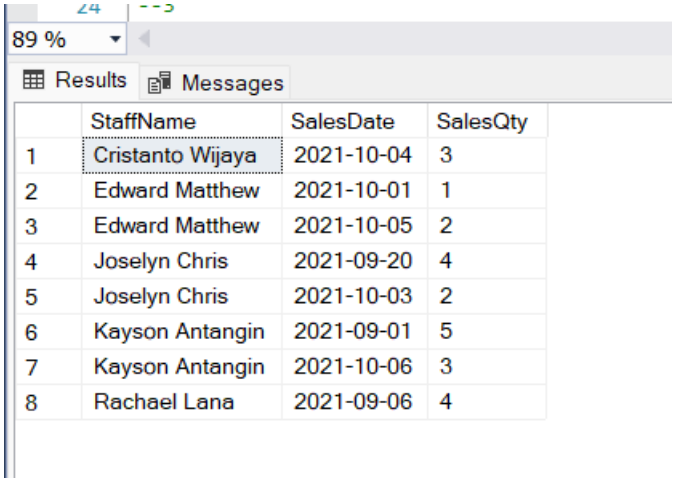
	StaffName	SalesDate	Total Sales
1	Kayson Antangin	01 Sep 2021	5
2	Cristanto Wijaya	05 Sep 2021	3
3	Rachael Lana	06 Sep 2021	4
4	Rachael Lana	08 Sep 2021	2
5	Anya Nyiaaa	09 Sep 2021	3
6	Anya Nyiaaa	11 Sep 2021	1
7	Kayson Antangin	17 Sep 2021	1
8	Joselyn Chris	20 Sep 2021	4
9	Darren Daviel	25 Sep 2021	2
10	Cristanto Wijaya	30 Sep 2021	3
11	Edward Matthew	01 Oct 2021	1
12	Joselyn Chris	03 Oct 2021	2
13	Cristanto Wijaya	04 Oct 2021	3
14	Edward Matthew	05 Oct 2021	2
15	Kayson Antangin	06 Oct 2021	3

Union :

```sql
--2
SELECT StaffName, SalesDate, SalesQty
FROM MsStaff ms
JOIN SalesTransaction st ON st.StaffID = ms.StaffID
JOIN SalesDetail sd ON sd.SalesID =  st.SalesID
WHERE sd.SalesQty > 3
UNION
SELECT StaffName, SalesDate, SalesQty
FROM MsStaff ms
JOIN SalesTransaction st ON st.StaffID = ms.StaffID
JOIN SalesDetail sd ON sd.SalesID =  st.SalesID
WHERE MONTH(SalesDate) = 10
```

Here I want to use union to Combine the validation In one table for the 1st iwant salesQty to be more than 3 and the 2nd I want month on sales date is on November and I combine it using UNION

Here's the result



89 %

Results | Messages

	StaffName	SalesDate	SalesQty
1	Cristanto Wijaya	2021-10-04	3
2	Edward Matthew	2021-10-01	1
3	Edward Matthew	2021-10-05	2
4	Joselyn Chris	2021-09-20	4
5	Joselyn Chris	2021-10-03	2
6	Kayson Antangin	2021-09-01	5
7	Kayson Antangin	2021-10-06	3
8	Rachael Lana	2021-09-06	4

3. The there are 4 characteristic data warehouse :

1. Subject – Oriented :

Subject-oriented data warehouses are those that analyze data based on organizationally-specific subjects rather than on specific applications or functions. The data warehouse is structured around the primary topics of the business (customers, products, and sales), rather than around areas of significant applications (customer invoicing, stock control and product sales). This is because a data warehouse is required to store data that is needed to support decisions, as opposed to data-oriented applications.

2. Time – Variant :

All data in the data warehouse can be said to be accurate or valid at any given time. To view the time interval used to measure the accuracy of a data warehouse

The way to use Time Variant :

1. The most straightforward way is to present the data warehouse at a specific time period, such as five to ten years in the future.

2. The second way is using a whole kind of variations or differences in time that are either implicitly or explicitly included in the data warehouse, an explicit inclusion of the element of time in days, weeks, months, and so forth. implicitly for instance, when the data is repeated quarterly or at the end of each month. The data will continue to contain an implicit reference to time.

3. The third method involved showing the time variation in the data warehouse through an extensive number of snapshots. S snapshot is a condensed view of the specific data that the user would want to see out of all the read-only data.

4. is The non volatile down below.

3. Integrated :

Data from multiple sources is combined in a data warehouse. Relational databases, flat files, structured and semi-structured data, metadata, cloud, and master data are a few examples of these.

When the sources are combined in a way that is stable, relatable, and ideally certifiable, provides business with confidence in the accuracy of the data.

4. Persistent and non – volatile :

Non-volatile refers to the fact that the data in the data warehouse is not updated in real time but rather on a regular basis when the operating system is refreshed. Instead of making a change to the database, the new data are being added as a supplement. New data is progressively added to the database as well as being continuously absorbed.

4. **Normalization Sales Transaction**

UNF :

BranchID, BranchLocation, BuyerID, BuyerName, BuyerEmail, PhoneNumber, BuyerAddress, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress, SalesID, TransactionDate, ToyID, ToyName,ToyStock, Quantity, ToySalesPrice, TotalPrice, TotalDiscount, PaymentType

1NF :

SalesHeader :

SalesID (PK), TransactionDate, BranchID, BranchLocation, BuyerID, BuyerName, BuyerEmail, PhoneNumber, BuyerAddress, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress

SalesDetail :

SalesID(PK FK), ToyID (PK), ToyName, ToyStock, Quantity, ToyPrice, PaymentType

2NF :

SalesHeader :

SalesID (PK), TransactionDate, BranchID, BranchLocation, BuyerID, BuyerName, BuyerEmail, BuyerPhoneNumber, BuyerAddress, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress

SalesDetail :

SalesID (PK FK), ToyID (PK FK), Quantity

MsToy :

ToyID (PK), ToyName, ToyStock, ToySalesPrice

3NF :

SalesHeader :

SalesID (PK), TransactionDate, **BranchID (FK), BuyerID (FK), StaffID (FK)**

SalesDetail :

SalesID(PK FK), ToyID (PK FK), Quantity

MsToy :

ToyID (PK), ToyName, ToyStock, ToySalesPrice

MsBranch :

BranchID (PK), BranchLocation

MsBuyer :

BuyerID (PK), BuyerName, BuyerEmail, BuyerPhoneNumber, BuyerAddress

MsStaff :

StaffID (PK), StaffName, StaffDOB, StaffEmail, StaffAddress

Normalization Purchase Transaction

UNF :

BranchID, BranchLocation, SupplierID, SupplierName, SupplierLocation, TransactionID, TransactionDate, ToyID, ToyName, ToyPurchasePrice, Quantity, TotalPurchase, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress

1NF :

PurchaseHeader :

TransactionID(PK), TransactionDate,BranchID, BranchLocation, SupplierID, SupplierName, SupplierLocation, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress

PurchaseDetail :

TransactionID (PK FK), ToyID (PK), ToyName, ToyPurchasePrice, Quantity

2NF :

PurchaseHeader :

SalesID (PK), TransactionDate, BranchID, BranchLocation, SupplierID, SupplierName, SupplierLocation, StaffID, StaffName, StaffDOB, StaffEmail, StaffAddress

PurchaseDetail :

SalesID (PK FK), ToyID (PK FK), Quantity

MsToy :

ToyID (PK), ToyName, ToyPurchasePrice

3NF :

Purchase Header :

TransactionID (PK), TransactionDate, BranchID(FK), SupplierID (FK), StaffID (FK)

PurchaseDetail :

TransactionID (PK FK), ToyID (PK FK), Quantity

MsToy :

ToyID (PK), ToyName, ToyPurchasePrice

MsBranch :

BranchID (PK), BranchLocation

MsSupplier :

SupplierID (PK), SupplierName, SupplierLocation

MsStaff :

StaffID (PK), StaffName, StaffDOB, StaffEmail, StaffAddress

Combine Sales Transaction And Purchase Transaction

Combine 3NF :

SalesHeader

SalesID (PK), TransactionDate, **BranchID (FK), BuyerID (FK), StaffID (FK)**

SalesDetail :

TransactionID (PK FK), ToyID (PK FK), Quantity

MsToy :

ToyID (PK), ToyName, ToyStock, ToySalesPrice, ToyPurchasePrice

MsBranch :

BranchID (FK), BranchLocation

MsBuyer :

BuyerID (PK), BuyerName, BuyerEmail, BuyerPhoneNumber, BuyerAddress

MsStaff :

StaffID (PK), StaffName, StaffDOB, StaffEmail, StaffAddress

PurchaseHeader :

TransactionID (PK), TransactionDate, **BranchID (FK), SupplierID (FK), StaffID (FK)**

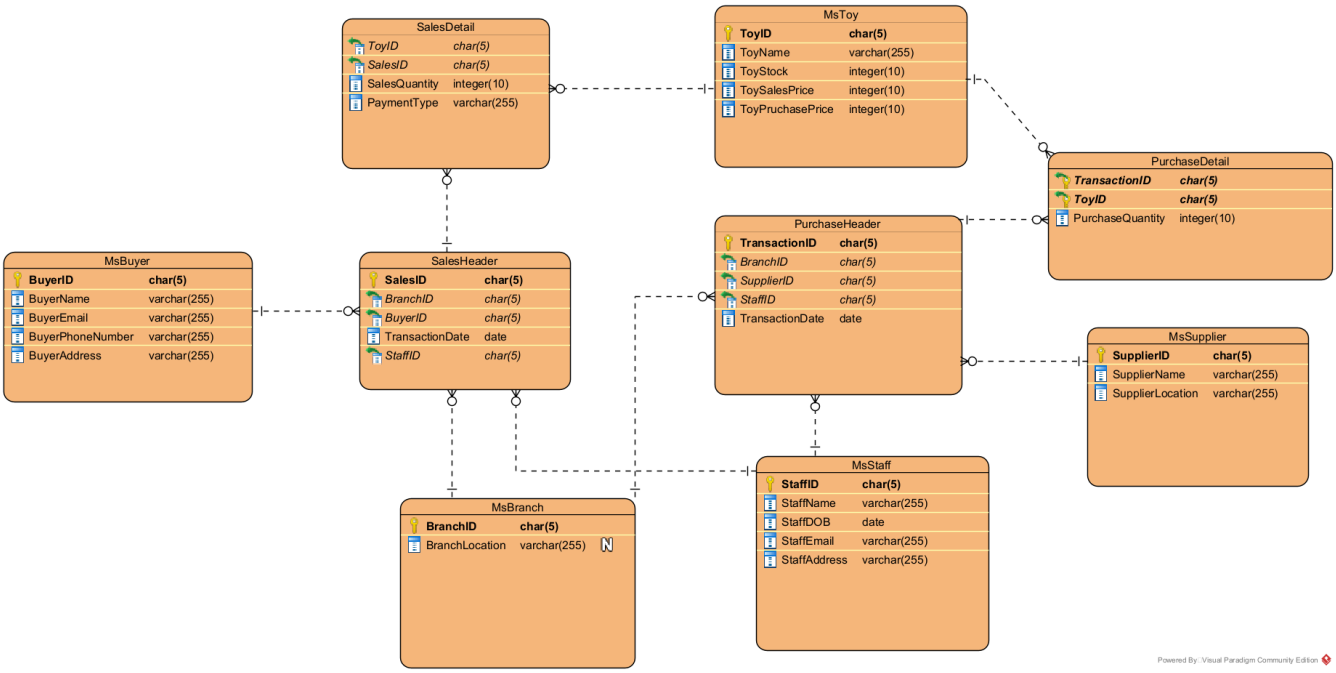
PurchaseDetail :

TransactionID (PK FK), ToyID (PK FK), Quantity

MsSupplier :

SupplierID (PK), SupplierName, SupplierLocation

5. ERD :



6.

```sql
--6
SELECT [Toy Type] = mt.ToyName , [Toy Sold] = SUM(sd.SalesQuantity),[Purchase Price] = mt.ToyPurchasePrice, [Sales Price] =
  mt.ToySalesPrice, [Profit] = ToySalesPrice - ToyPurchasePrice, [Profitable Percentage] = CAST(floor((ToySalesPrice -
  ToyPurchasePrice) / CONVERT(NUMERIC(18,2),ToyPurchasePrice) * 100) AS VARCHAR) + '%'
FROM MsToy mt
JOIN SalesDetail sd ON sd.ToyID = mt.ToyID
JOIN SalesHeader sh ON sh.SalesID = sd.SalesID
WHERE MONTH(SalesTransactionDate) BETWEEN 1 AND 3 AND YEAR (SalesTransactionDate) = 2022
GROUP BY ToyName, ToyPurchasePrice, ToySalesPrice
```

Here's the code for my attempt doing question number 6, as you can see the profitable percentage I use CAST so I can combine the number with %, but it can also use 'CONCAT' but the difference is CAST is just to switch from other form such as INT TO VARCHAR, but if we use the CONCAT the number will remain INT.

I also use SUM to combine all the same items into one quantity so we can see how many each Toy Type That has been sold, this FUNCTION needs to be Followed by GROUP BY function

I use FLOOR to round the number, so if 33.333% it goes to 33%

Here's the result

Results Messages

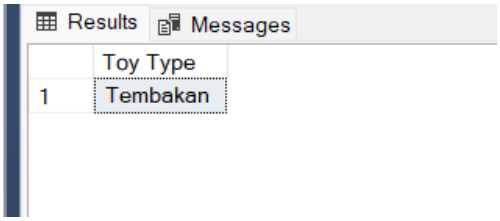
	Toy Type	Toy Sold	Purchase Price	Sales Price	Profit	Profitable Percentage
1	Figurin	10	95000	150000	55000	57%
2	layangan	20	5000	15000	10000	200%
3	Robot	7	45000	60000	15000	33%
4	Tembakan	0	15000	50000	35000	233%

7.

```sql
--7
SELECT [Toy Type] = mt.ToyName
FROM MsToy mt
JOIN SalesDetail sd ON sd.ToyID = mt.ToyID
JOIN SalesHeader sh ON sh.SalesID = sd.SalesID
WHERE mt.ToyID IN (
    SELECT sd.ToyID
    FROM SalesDetail sd
    WHERE mt.ToyStock - sd.SalesQuantity = mt.ToyStock
) AND MONTH(SalesTransactionDate) BETWEEN 1 AND 3 AND YEAR (SalesTransactionDate) = 2022
```

Here's the code for attempting the 7 questions, The question needs to be done using subquery and the question want me to show Toys Type that haven't been sold in January to march 2022, so I choose the IN function then I use the validation and it goes like this : if the stock of the toys – by What has been sold the amount still the same then show it, then validation for the date and it's the same as in question 6

Here's the Result :



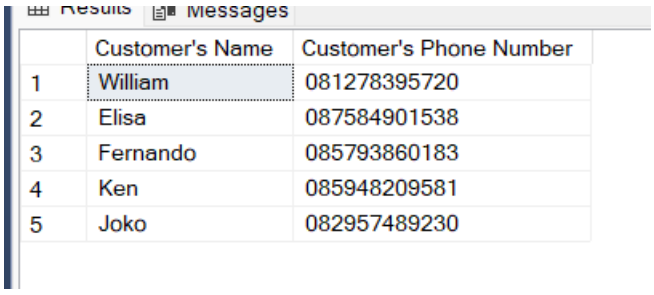
Results Messages

	Toy Type
1	Tembakan

8.

```sql
--8
CREATE VIEW ViewCustomer
AS
SELECT [Customer's Name] = mb.BuyerName, [Customer's Phone Number] = mb.buyerPhone
FROM MsBuyer mb
```

Here's my attempt to do question number 8, in number 8 is simple i just needs to create a view and display the customer name and the phone number and here's the result :



Results Messages

	Customer's Name	Customer's Phone Number
1	William	081278395720
2	Elisa	087584901538
3	Fernando	085793860183
4	Ken	085948209581
5	Joko	082957489230

VIDEO LINK :

https://youtu.be/1lzCNXw8W_Y