```
--- LESSON 1 -- SIMPLE QUERY

-- Đây là câu chú thích

/* Chú thích mà có nhiều dòng
dòng 1
dòng 2
dòng 3
*/

-- I. Hiển thị dữ liệu
-- Lệnh: SELECT

SELECT 'Đây là tên của tui'

SELECT N'Đây là tên của tui'

-- Hiển thị dữ liệu từ bảng trong database

SELECT * -- hiển thị tất cả dữ liệu trong bảng
FROM SalesLT.Customer

-- 13552 dòng

-- Hiển thị 1 vài columns
SELECT CustomerID
        , Title
        , FirstName
        , LastName
        , MiddleName
FROM SalesLT.Customer
ORDER BY LastName ASC , CustomerID DESC -- ascending, DESC : descending

-- FILTERING DATA: Chọn lọc dữ liệu
-- WHERE :

SELECT CustomerID
        , Title
        , FirstName
        , LastName
        , MiddleName
FROM SalesLT.Customer
WHERE CustomerID < 100

/* Exercise 2: Write a query using a WHERE clause that displays all the products
listed
in the SalesLT.Product table which have the color “black” or ‘red’ and size is 'S'.
Display the ProductID, Name, Color, Size, for each one. */

SELECT *
FROM SalesLT.Product

SELECT ProductID
        , Name
        , Color
        , Size
FROM SalesLT.Product
WHERE ( Color = 'Black' OR Color = 'Red')
        AND Size = 'S'

-- đáp án: 5 dòng

-- IN: so sánh với giá trị trong 1 tập hợp
SELECT ProductID
        , Name
        , Color
   , Size
FROM SalesLT.Product
WHERE Color  IN ('Black', 'Red')
        AND Size = 'S'

/* Exercise 3: Retrieve the ProductID, ProductNumber and Name of the products, that
must
- Have Product number begins with ‘FR-‘
- Have Product name contains ‘HL’ or ‘Mountain’ */

SELECT ProductID
        , ProductNumber
        , Name
FROM SalesLT.Product
```

```
WHERE ProductNumber LIKE 'FR-%'
AND ( Name LIKE '%HL%' OR Name LIKE '%Mountain%' )

-- đáp án: 47 dòng

SELECT ProductID
        , ProductNumber
        , Name
FROM SalesLT.Product
WHERE ProductNumber LIKE '________%' -- tìm ra các product number có ít nhất 8
kí tự
-- AND Name LIKE '[a-d]%'  -- tìm các name có kí tự đầu tiên nằm trong dãy a,b,c,d
AND Name LIKE '[^a-d]%'  -- tìm các name có kí tự đầu tiên khác a,b,c,d
```

-- LESSON 2: BUILT IN FUNCTION

-- DATA TYPE

-- STring
--- char(n) : char(5) --> phai co du 5 ki tu
 trang , trinh, huong còn hoa thì ko thoả mãn
--- varchar(n): cho phep toi da n ki tu:
varchar(5): hieu, trinh, hoa, (là tối đa 5 kí tự)
--- nvarchar(5): cho phep ki tu dac biet và tối đa 5 kí tự ví dụ kí tự đặc biệt như là dấu ở các chữ viết của việt nam
...

 -- Ghep chuoi bang phep +

```
 SELECT 'toi ten ' + 'hieu'
 SELECT 'toi ten ' + 1
```
(có thể thay việc ghép bằng dấu + bằng biểu thức concat)
AS: alias

-- Cach Naming trong code :
--- Snake: ho_va_ten
--- Camel: hoVaTen
--- pacal: HoVaTen

```
-- Thứ tự execution của SQL:
-- FROM -> WHERE -> SELECT (DISTINCT) -> ORDER BY -> LIMIT(TOP N
[PERCENT])

SELECT CustomerID
	, FirstName
	, MiddleName
	, LastName
	, ISNULL(MiddleName, '-') AS new_middle_name
	, FirstName + ' '+  ISNULL(MiddleName, '-') + ' '+ LastName AS full_name
	, COALESCE(MiddleName,FirstName,LastName) AS new_name_2
FROM SalesLT.Customer
ORDER BY new_middle_name ASC

--  WHERE MiddleName IS NOT NULL

-- DATE TIME FUNCTION

SELECT SalesOrderID
	, OrderDate
	, ShipDate
	-- , DAY(OrderDate) AS day ( triết xuất ngày trong cột)
	-- , MONTH(OrderDate) AS month
	-- , YEAR(OrderDate) AS year
	-- , DATEPART(day, OrderDate) AS day_1 ( triết xuất ngày trong cột
orderdate)
	, CURRENT_TIMESTAMP AS [current_time] ( triết xuất thời gian hiện tại)
	GETDATE() AS [current_time] (cũng là triết xuất ngày hiện tại)
	, DATEADD( hour, 7, CURRENT_TIMESTAMP) AS vn_time ( cộng thêm 7
giờ cho cột thời gian hiện tại để đưa về thời gian hiện tại của việt nam )
	, DATEDIFF( day, OrderDate, ShipDate) AS deliver_day ( lấy ngày của cột
orderdate trừ cho ngày trong cột shipdate )
FROM SalesLT.SalesOrderHeader

-- STRING FUNCTIONS

SELECT CustomerID
	, LastName
	, CompanyName
	, LEN(CompanyName) AS lenght (tính độ dài chuỗi trong cột)
```

```
        , LEFT(CompanyName, 5) AS left_name ( lấy 5 kí tự bên trái của cột
companyname)
        , RIGHT(CompanyName, 5) AS right_name
        , CHARINDEX('B', CompanyName) AS position (kết quả trả ra số thứ tự kí tự
B trong cột companyname) (vd trang thì charindex của kí tự a sẽ trả kết quả là 3)
        , SUBSTRING(CompanyName, 3, 4) AS new_string( triết xuất dữ liệu số thứ
tự thứ 3 trong các hàng của cột companyname và lấy sau đó 4 kí tự)
        , REPLACE (CompanyName, 'Bike', 'Car') AS new_replace ( là hàm để thay
thế)
   , REVERSE(CompanyName) ( hàm đảo ngược vd thanh thì chuyển thành hnaht)
FROM SalesLT.Customer


/* Exerise 6: From table SalesLT.Customer
Get name of each sale man. Name is last part of SalesPerson.
Example: adventure-works\jun0 -> Name = jun0 */


SELECT * FROM SalesLT.Customer

-- Huong:
Select salesPerson
        , CHARINDEX ('\', salesPerson)
        , RIGHT(salesPerson,LEN(salesPerson)-CHARINDEX ('\', salesPerson)) AS
name
From SalesLT.Customer

-- Nguyen
SELECT DISTINCT salesPerson, REPLACE(SalesPerson,'adventure-works\','') AS
[SalesPerson]
FROM SalesLT.Customer

-- Minh Duc
SELECT DISTINCT SUBSTRING(SalesPerson, CHARINDEX ('\', salesPerson) +
1,100) AS NAME
FROM SalesLT.Customer
-- An
SELECT SUBSTRING (SalesPerson, CHARINDEX ('\', salesPerson) +1,
LEN(SalesPerson)-LEN(LEFT(SalesPerson,16))) as Name
FROM SalesLT.Customer
```

```
-- LOGICAL FUNCTIONS

-- CASE WHEN
SELECT ProductID
        , Name
        , ListPrice
        , CASE WHEN ListPrice < 100 THEN 'Thap'
        WHEN ListPrice >= 100 AND ListPrice < 500 THEN 'Trung binh'
        ELSE 'Cao'
        END AS group_price
FROM SalesLT.Product
WHERE (CASE WHEN ListPrice < 100 THEN 'Thap'
        WHEN ListPrice >= 100 AND ListPrice < 500 THEN 'Trung binh'
        ELSE 'Cao'
        END) = 'Thap'( trong mệnh đề where này ko dc sử dụng group_price vì mệnh
đề select chạy sau mệnh đề where nên là cột này chưa dc tạo)

-- Case when co the mang vao WHERE

?: Lam sao de hien thi cac dong group_price la 'Thap'

-- IIF:

SELECT IIF ( 50 < 20, 'TRUE', 'FALSE' ) AS RESULT (là hàm giống case when then
nhưng nó chỉ áp dụng cho 2 điều kiện nếu đúng thì nó trả về biểu thức 1 nếu sai nó trả
về biểu thức 2 )

SELECT ListPrice
        , IIF( ListPrice < 100, 'Thap', 'Cao') AS result
FROM SalesLT.Product

/* Exercise 5: Retrieve shipping status
You have been asked to create a query that returns a list of sales order IDs and order
dates with
a column named ShippingStatus that contains the text Shipped for orders with a
known ship date,
and Awaiting Shipment for orders with no ship date. */

SELECT SalesOrderID
```

```
        , ShipDate
        , CASE WHEN ShipDate IS NOT NULL THEN 'Shipped'
        ELSE 'Awaiting Shipment'
        END AS ShippingStatus
        , IIF(ShipDate IS NOT NULL, 'Shipped', 'Awaiting Shipment') AS
ShippingStatus_2
FROM SalesLT.SalesOrderHeader


-- CONVERTION FUNCTION : ham chuyen doi kieu du lieu
-- CAST (column AS new_data_type)
-- CONVERT ( data_type, column, style) (styple = 101, 102.. có bảng trên chỗ convert
của microsoft nó là các kiểu hình thức khác nhau cho kiểu dữ liệu)

SELECT SalesOrderID
        , OrderDate
        , 'ma don hang: ' + CAST (SalesOrderID AS varchar) AS new_id
        , CONVERT (varchar, OrderDate, 102) AS new_time
        , CONVERT (decimal(10,2), SalesOrderID) AS new_id_2
FROM SalesLT.SalesOrderHeader

-- LESSON 3: JOIN & UNION --

-- JOIN: Ghep bang du lieu --> Mo rong du lieu theo chieu ngang

--- INNER JOIN:
--- Syntax:

SELECT
FROM table1 AS t1
INNER JOIN table2 AS t2
ON t1.column_a = t2.column_b

SELECT TOP 5 * FROM SalesLT.Customer
SELECT TOP 5 * FROM SalesLT.CustomerAddress
--> Related column: CustomerID

SELECT * -- hien thi tat ca cot
FROM SalesLT.Customer AS cus
INNER JOIN SalesLT.CustomerAddress AS cus_address
```

```
ON cus.CustomerID = cus_address.CustomerID

SELECT cus.CustomerID
   , cus_address.CustomerID
   , LastName
        , Phone
        , AddressID
   , AddressType
FROM SalesLT.Customer AS cus
INNER JOIN SalesLT.CustomerAddress AS cus_address
ON cus.CustomerID = cus_address.CustomerID
-- Dap an: 417 rows ?

SELECT  * FROM SalesLT.Customer -- 847 rows --> 847 khach hang

SELECT  DISTINCT CustomerID  FROM SalesLT.CustomerAddress -- 417 rows -->
407 khach hang
--> co 10 khach hang co 2 dia chi

SELECT CustomerID
        , COUNT(AddressID) AS number_add
FROM SalesLT.CustomerAddress
GROUP BY CustomerID
ORDER BY number_add DESC

-- LEFT JOIN:
SELECT cus.CustomerID AS cus_id
        , cus_address.CustomerID AS cus_add_id
        , LastName
        , Phone
        , AddressID
   , AddressType
FROM SalesLT.Customer AS cus --> 847 rows : 847 khach hang
LEFT JOIN SalesLT.CustomerAddress AS cus_address --> 417 rows , 407 khach
hang
ON cus.CustomerID = cus_address.CustomerID
-- WHERE cus.CustomerID = 29559
-- Dap an: 857 rows (nhieu hon 10 dong)

-- RIGHT JOIN:
```

```
SELECT cus.CustomerID AS cus_id
        , cus_address.CustomerID AS cus_add_id
        , LastName
        , Phone
        , AddressID
   , AddressType
FROM SalesLT.Customer AS cus --> 847 rows : 847 khach hang
RIGHT JOIN SalesLT.CustomerAddress AS cus_address --> 417 rows , 407 khach
hang
ON cus.CustomerID = cus_address.CustomerID
-- Dap An: 417 rows --> tat ca khach hang trong CustomerAddress deu ton tai trong
bang Customer

-- FULL JOIN:
SELECT cus.CustomerID AS cus_id
        , cus_address.CustomerID AS cus_add_id
        , LastName
        , Phone
        , AddressID
   , AddressType
FROM SalesLT.Customer AS cus --> 847 rows : 847 khach hang
FULL JOIN SalesLT.CustomerAddress AS cus_address --> 417 rows , 407 khach
hang
ON cus.CustomerID = cus_address.CustomerID

-- Dap An: 857 rows giong LEFT JOIN vi tat ca ID trong bang CustomerAddress deu
thuoc Customer

/* Exercise 6: Generate invoice reports
Adventure Works Cycles sells directly to retailers, who must be invoiced for their
orders. You
have been tasked with writing a query to generate a list of invoices to be sent to
customers.
Retrieve customer orders:
As an initial step towards generating the invoice report, write a query that returns the
company name from the SalesLT.Customer table, and the sales order ID and total due
from
the SalesLT.SalesOrderHeader table. */

SELECT TOP 5 * FROM SalesLT.Customer
SELECT  * FROM SalesLT.SalesOrderHeader

-- B1: Xac dinh cac tables:
Customer: -- dimension
SalesOrderHeader: -- fact
-- B2: Xac phep JOIN va Related column: dung JOIN cung dc, CustomerID
-- B3: Viet JOIN ...

-- anh Huy
SELECT cus.CustomerID
        , CompanyName
        , SalesOrderID
        , TotalDue
FROM SalesLT.Customer AS cus
INNER JOIN SalesLT.SalesOrderHeader AS sales_order
ON cus.CustomerID = sales_order.CustomerID
-- 32 rows:

-- Huong
SELECT CompanyName
        , SalesOrderID
   , TotalDue
FROM SalesLT.Customer AS cus
RIGHT JOIN SalesLT.SalesOrderHeader AS order_header
ON cus.CustomerID = order_header.CustomerID
-- 32:

-- An
SELECT CompanyName
        , SalesOrderID
        , TotalDue
FROM SalesLT.Customer as cus
RIGHT JOIN SalesLT.SalesOrderHeader as header
ON cus.CustomerID = header.CustomerID

-- Hung:
SELECT CompanyName
        , SalesOrderID
   , TotalDue
FROM SalesLT.Customer as cus
```

```
RIGHT JOIN SalesLT.SalesOrderHeader AS s_o_header
ON cus.CustomerID = s_o_header.CustomerID

SELECT cus.CustomerID, CompanyName
        , SalesOrderID
   , TotalDue
FROM SalesLT.Customer as cus
LEFT JOIN SalesLT.SalesOrderHeader AS s_o_header
ON cus.CustomerID = s_o_header.CustomerID

-- Giua LEFT va RIGHT ?
---> Nen dung 1 trong 2: va nen la LEFT
SELECT CompanyName
        , SalesOrderID
   , TotalDue
FROM SalesLT.SalesOrderHeader AS s_o_header -- fact
LEFT JOIN SalesLT.Customer AS cus -- dim
ON s_o_header.CustomerID = cus.CustomerID

-- Thu tu SQL: FROM --> JOIN --> WHERE --> SELECT --> ORDER BY -->
LIMIT

-- Tip1: Minh thuong xuat phat phat tu FACT (bang chua du lieu minh phan phan tich,
nhieu du)

/* Exercise 7: Write a query using SalesLT.ProductCategory and SalesLT.Product,
display ProductID, ProductName, Color and ProductCategoryID of product
which ProductCategoryName contains 'Mountain' */

-- B1: cac tables
SELECT *  FROM SalesLT.ProductCategory -- Dim --> Nganh hang (41 dong --> 41
nganh hang)
SELECT *  FROM SalesLT.Product --> Dim --> San pham (295 san pham)
-- b2: Phep JOIN nao
Giua 2 bang DIM --> Xuat phat tu bang co level nho hon (chi tiet va nhieu dong du
lieu hon)
-- B3: Viet JOIN

-- Duc:
select ProductID
```

```
        ,product.Name
   ,Color
   ,product_category.ProductCategoryID
from SalesLT.ProductCategory  as product_category  -- dim category 41 dong
left join SalesLT.Product as product                    --- dim
on product_category.ProductCategoryID = product.ProductCategoryID
where product_category.Name like '%Mountain%'

-- Tram
SELECT
        product.ProductID
   , product.Name
   , product.Color
   , product.ProductCategoryID
FROM
(SELECT
   ProductCategoryID
        , ProductID
   , Name
   , Color
FROM SalesLT.Product) AS product -- Truy Van long ghep --> buoi hoc sau
INNER JOIN
(SELECT
        ProductCategoryID
FROM SalesLT.ProductCategory
WHERE Name LIKE '%Mountain%' ) AS product_cate
ON product.ProductCategoryID=product_cate.ProductCategoryID

/* Exercise 8: Write a query using SalesLT.SalesOrderHeader, SalesLT.Product and
SalesLT.SalesOrderDetail display SalesOrderID, SalesOrderDetailID, ProductID,
ProductName, OrderDate, LineTotal, SubTotal */

-- B1: Xac dinh cac tables
SELECT TOP 5 *  FROM SalesLT.SalesOrderHeader --> fact header thoi gian giao
hang
SELECT TOP 5 *  FROM SalesLT.Product -- dim product
SELECT TOP 5 *  FROM SalesLT.SalesOrderDetail --> Fact trung tap
-- Bang Detail --> Header: SalesOrderID
-- Bang Detail --> Product: ProductID
-- B2: LEFT detail --> Product --> Header:
```

```
-- B3:

SELECT detail.SalesOrderID
        , SalesOrderDetailID
        , detail.ProductID
        , Name
        , OrderDate
        , LineTotal
        , SubTotal
FROM SalesLT.SalesOrderDetail AS detail
LEFT JOIN SalesLT.Product AS pro
   ON detail.ProductID = pro.ProductID
LEFT JOIN SalesLT.SalesOrderHeader AS header
   ON detail.SalesOrderID = header.SalesOrderID


-- Kết nối qua database mới: PayTM

SELECT * FROM dim_payment_channel
SELECT * FROM dim_platform
SELECT * FROM dim_scenario
SELECT * FROM dim_status

SELECT TOP 5 * FROM fact_transaction_2019 -- data của 2019
SELECT TOP 5 * FROM fact_transaction_2020 -- data của 2020
--> Optimize luu va truy van

SELECT customer_id
        , transaction_id
        , transaction_time
        , payment_platform
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_platform AS plat
ON fact_19.platform_id = plat.platform_id

SELECT * FROM fact_transaction_2019 -- 396K
SELECT * FROM fact_transaction_2020 -- 801k

-- UNION : Ghep bang chieu doc
```

```
SELECT *
FROM fact_transaction_2019
UNION -- Gop lai va loai bo Trung Lap
SELECT *
FROM fact_transaction_2020
--> 1198K dong

SELECT TOP 1000 customer_id
        , transaction_id
FROM fact_transaction_2019
UNION ALL
SELECT TOP 1000 customer_id
        , transaction_id
FROM fact_transaction_2020

-- dap 2000 dong

-- UNION: dung khi ghep 2 bang cung tinh chat, format

-- THU TU SQL: FROM --> JOIN --> WHERE --> SELECT --> ORDER BY -->
LIMIT --> UNION
-- LESSON 4: SUBQUERIES - CTE - GROUP BY

--- Subqueries (Mệnh đề truy vấn lồng ghép)

--- Xuất hiện ở lệnh SELECT: chỉ trả ra 1 giá trị

 SELECT TOP 10
        transaction_id
        , customer_id
        , original_price - discount_value AS charged_amount_cus
        , (SELECT MIN(charged_amount) FROM fact_transaction_2020) AS
min_amount -- có thể lấy ra kết từ bảng khác
        , (SELECT MAX(charged_amount) FROM fact_transaction_2020) AS
max_amount
        , (SELECT TOP 2 charged_amount FROM fact_transaction_2020) AS top_1
FROM fact_transaction_2019  -- table chinh


-- FROM : Phải naming lại cho cái bảng trung gian đó
```

```
SELECT *
FROM
	( SELECT TOP 1000 *
	FROM fact_transaction_2019
	UNION
	SELECT TOP 1000 *
	FROM fact_transaction_2020 ) AS fact_table -- Subquery la tao ra bang trung
LEFT JOIN dim_platform AS platform
	ON fact_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
	ON fact_table.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH (transaction_time) = 1

-- WHERE : tạo ra 1 giá trị hoặc 1 tập hợp để so sánh

SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2020 AS fact_20
LEFT JOIN dim_platform AS platform
	ON fact_20.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
	ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
	AND payment_platform = 'ios'
	AND customer_id IN ( SELECT DISTINCT customer_id
			FROM fact_transaction_2019
			WHERE MONTH(transaction_time) = 1) -- Subquery

-- II. CTE: Common Table Expession: Tạo bảng tạm tồn tại chỉ trong câu lệnh truy
vấn
-- Syntax:
WITH table_name AS (
	SELECT
	FROM ...
)
SELECT *
FROM table_name

 -- ex1: Ví dụ o tren
```

```
SELECT *
FROM
        ( SELECT TOP 1000 *
        FROM fact_transaction_2019
        UNION
        SELECT TOP 1000 *
        FROM fact_transaction_2020 ) AS fact_table -- Subquery tạo bảng trung gian
LEFT JOIN dim_platform AS platform
        ON fact_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_table.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'web' AND MONTH (transaction_time) = 1

-- CTE:
WITH fact_table AS (
  SELECT TOP 1000 *
        FROM fact_transaction_2019
        UNION
        SELECT TOP 1000 *
        FROM fact_transaction_2020
)
SELECT *
FROM fact_table
LEFT JOIN dim_platform AS platform
        ON fact_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_table.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'web' AND MONTH (transaction_time) = 1



-- ex 2: Tạo nhiều nhiều bảng tạm bằng CTE --> dc phep tach nho cau truy van thanh
buoc , nhieu bang
WITH fact_table AS (
        SELECT TOP 1000 *
        FROM fact_transaction_2019
        UNION
        SELECT TOP 1000 *
        FROM fact_transaction_2020
)
```

```
, success_table AS ( -- buoc 2: loc du lieu
        SELECT *
        FROM fact_table
        WHERE status_id = 1 AND MONTH(transaction_time) = 1
)
SELECT customer_id , transaction_id
FROM success_table
LEFT JOIN dim_platform AS platform
        ON success_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON success_table.payment_channel_id = channel.payment_channel_id

SELECT *
FROM success_table --> Bang tao boi CTE chi ton tai trong chinh cau truy van do
thoi.


-- Chúng ta có quyền tạo nhiều bảng trung gian bằng CTE , và bảng trung gian phía
sau có thể sử dụng
-- kết quả từ bảng trung gian thứ 1

-- GROUP BY: Gom nhóm để Tính toán theo từng nhóm đối tượng

SELECT TOP 10 * FROM fact_transaction_2019 -- 396K dong

-- ex 3: Hãy thống kê xem mỗi khách hàng đã thanh toán bao nhiêu giao dịch nam
2019?

SELECT customer_id -- hien thi column ma GROUP BY
        , COUNT (transaction_id) AS number_trans
FROM fact_transaction_2019
GROUP BY customer_id

-- 30,130 dong --> So luong khach hang



/* ex 4: Hãy thống kê xem mỗi khách hàng trong nam 2019:
- đã thanh toán bao nhiêu giao dịch thanh cong?
- số ngày phát sinh giao dịch thanh cong (active days) của mỗi khách hàng là bao
nhiêu? */

SELECT customer_id
        , COUNT (transaction_id) AS number_trans -- dem so dong
        , COUNT ( DISTINCT CONVERT (varchar, transaction_time, 101) ) AS
number_days -- truoc khi phai chuyen format ve yyyy/mm/dd
FROM fact_transaction_2019 fact_19
LEFT JOIN dim_status sta
ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
GROUP BY customer_id

SELECT customer_id
        , COUNT (status_id) AS success_number -- COUNT nó sẽ đếm số dòng
FROM fact_transaction_2019
WHERE status_id = 1 -- giao dich thanh cong
        AND customer_id = 2917
GROUP BY customer_id




/* Excerise 5:
Retrieve a report that includes: total number of transactions, number of customers
and total amount by each month in 2019.
Only show the results of these transactions had status is successful */
-- Cố gắng hình dung output có những columns nào?
-- total number of transactions: đếm giao dịch
-- number of customers: đếm số người
-- total amount: tổng số tiền
-- by each month: MONTH(transaction_time)

WITH temp_table AS (
SELECT customer_id, transaction_id, charged_amount, transaction_time
        , MONTH (transaction_time) AS month
FROM fact_transaction_2019
WHERE status_id = 1
)
```

```
SELECT month
        , COUNT (transaction_id) AS number_trans
        , COUNT (DISTINCT customer_id) AS number_customer
        , SUM (CAST (charged_amount AS FLOAT)) AS total_amount
        -- , SUM (1.0* charged_amount) AS total_amount
FROM temp_table
GROUP BY month
ORDER BY month



/* Excerise 6:
Retrieve a report that includes: total number of transactions, number of customers and
total amount
by each month, each category in 2019.
Only show the results of these transactions had status is successful */

-- b1: JOIN và xử lý điều kiện:
WITH temp_table AS (
SELECT customer_id, transaction_id, charged_amount, transaction_time, category
        , MONTH (transaction_time) AS month
FROM fact_transaction_2019 fact_19
LEFT JOIN dim_scenario scena
ON fact_19.scenario_id = scena.scenario_id
WHERE status_id = 1
) -- b2: Gom nhóm theo month và category
SELECT month, category
        , COUNT (transaction_id) AS number_trans
        -- , COUNT (DISTINCT customer_id) AS number_customer
        , SUM (CAST (charged_amount AS FLOAT)) AS total_amount
FROM temp_table
GROUP BY month, category
ORDER BY month

-- LESSON 6: WINDOW FUNCTION --

-- 1. Đếm số lượng giao dịch theo tháng bằng Window function


SELECT DISTINCT
```

```
        MONTH (transaction_time) AS month
        , COUNT (transaction_id) OVER ( PARTITION BY MONTH
(transaction_time)) AS number_trans_month
FROM fact_transaction_2019

-- RANKING FUNCTION: Hàm xếp hạng

/*
Exercise 22: part 1
Đánh giá tốc độ tăng trưởng theo từng tháng của sản phẩm Telecom (chỉ tính các giao
dịch thành công)
thông qua các chỉ số:
- Số lượng giao dịch
- Số lượng khách hàng
- Tổng số tiền

--- part 2:
Sau đó hãy hiển thị thêm các columns sau:
- accummulated_number_trans: Tổng số lượng giao dịch cộng dồn theo tháng
- accummulated_number_customer: Tổng số khách hàng cộng dồn theo tháng
- accummulated_total_amount: Tổng số tiền cộng dồn theo tháng
*/

-- part 1:
-- cách 1 : GROUP BY
WITH summary_month AS (
SELECT MONTH(transaction_time) AS month
        , COUNT (transaction_id) AS number_trans
        , COUNT (DISTINCT customer_id) AS number_customer
        , SUM (1.0* charged_amount) AS total_amount
FROM fact_transaction_2019
GROUP BY MONTH(transaction_time)
) -- part 2
SELECT *
        , SUM (number_trans) OVER (ORDER BY month ASC ) AS
accummulating_trans
        , SUM (number_customer) OVER (ORDER BY month ASC ) AS
accummulating_customer
   , SUM (total_amount) OVER (ORDER BY month ASC ) AS
accummulating_amount
```

```
          , SUM (number_customer) OVER () AS total_customer
FROM summary_month

-- Ví dụ thêm:
-- Tạo 1 column tính tổng số khách hàng của cả năm
-- Tính số lượng khách hàng theo từng H1, H2

WITH summary_month AS (
SELECT IIF (MONTH(transaction_time) <7, 'H1', 'H2') AS half_year
          , MONTH(transaction_time) AS month
          , COUNT (transaction_id) AS number_trans
          -- , COUNT (DISTINCT customer_id) AS number_customer
          -- , SUM (1.0* charged_amount) AS total_amount
FROM fact_transaction_2019
GROUP BY IIF(MONTH(transaction_time) <7, 'H1', 'H2'),
          MONTH(transaction_time)
)
SELECT *
          , SUM (number_trans) OVER (ORDER BY month ASC ) AS
accummulating_trans_year
          , SUM (number_trans) OVER (PARTITION BY half_year ORDER BY month
ASC) accummulating_trans_half_year
          , SUM (number_trans) OVER () AS total_trans_year
          , SUM (number_trans) OVER (PARTITION BY half_year) AS
total_trans_h1_h2
FROM summary_month

/*
Exercise 22: part 1
Đánh giá tốc độ tăng trưởng theo từng tháng
thông qua các chỉ số:
- Số lượng giao dịch
- Số lượng khách hàng
- Tổng số tiền */

--- Tính part 1 bằng WINDOW FUNCTION
-- DISTICT không apply vào WINDOW FUNCTION
WITH rank_cus AS (
SELECT
month (transaction_time) AS month
```

```
, count (transaction_id) OVER (PARTITION BY month(transaction_time)) as
total_trans
, DENSE_RANK() OVER (PARTITION BY month(transaction_time) ORDER BY
customer_id) as rank_customer
, sum(charged_amount*1.0) OVER (PARTITION BY month(transaction_time))
total_amount
FROM fact_transaction_2019 AS fact_19
JOIN dim_scenario AS scena
ON fact_19.scenario_id = scena.scenario_id
WHERE category = 'Telco')

SELECT DISTINCT month, total_trans, total_amount
        , MAX(rank_customer) OVER (PARTITION BY month) AS number_customer
FROM rank_cus
ORDER BY month

/* Exercise 23: Dựa trên ví dụ bài 22 (tính số lượng khách hàng theo tháng)
Hãy đánh giá yếu tố số lượng khách hàng theo từng tháng của năm 2020 tăng hay
giảm bao nhiêu %
so với cùng kì năm trước. (Tức tháng 1/2020 tăng trưởng bao nhiêu % so với tháng 1
năm 2019)
*/

-- b1: Đếm số khách hàng theo tháng của 2 năm (2019 và 2020)
WITH summary_month AS (
SELECT YEAR (transaction_time) AS YEAR
        , MONTH(transaction_time) AS month
        , COUNT (DISTINCT customer_id) AS number_customer
FROM ( SELECT * FROM fact_transaction_2019
        UNION
        SELECT * FROM fact_transaction_2020) AS total_fact
JOIN dim_scenario AS scena
ON total_fact.scenario_id = scena.scenario_id
WHERE category = 'Telco'
GROUP BY YEAR (transaction_time), MONTH(transaction_time)
) -- b2: Tìm số lượng khách hàng cùng năm trước?
, previous_table AS
( SELECT *
        , LAG(number_customer, 12) OVER (ORDER BY year ASC, month ASC) AS
previous_result
```

```
FROM summary_month
)  -- b3: tính tỉ lệ tăng trưởng = kì hiện tại/kì trước - 1
SELECT *
        , FORMAT ( 1.0*number_customer/previous_result - 1, 'p') AS diff_pct
FROM previous_table

-- LEAD (column_value, N)

/* Exercise 24: Tính khoảng cách trung bình giữa các lần thanh toán theo từng khách
hàng trong nhóm Telecom. */

WITH previous_table AS (
SELECT customer_id
        , transaction_id
        , transaction_time
        , previous_time = LAG (transaction_time, 1) OVER (PARTITION BY
customer_id ORDER BY transaction_time)
FROM fact_transaction_2019 AS total_fact
JOIN dim_scenario AS scena
ON total_fact.scenario_id = scena.scenario_id
WHERE category = 'Telco'
)
, gap_table AS (
SELECT *
        , DATEDIFF (day,previous_time, transaction_time ) AS gap_day
FROM previous_table
)
SELECT customer_id
        , AVG(gap_day) AS avg_time
FROM gap_table
GROUP BY customer_id
HAVING AVG(gap_day) IS NOT NULL
```

-- Query Note Lesson 9 --

-- 1. Tạo database :

Cú pháp : CREATE DATABASE name

/*

Tạo Database tên là napas_transaction

```
Tiếp theo tạo các bảng dữ liệu lưu trư thông tin sau:
1. Bảng transaction bao gồm các thông tin:
(trans_id, customer_id, trans_type, bank_id, amount, trans_status, trans_date)
2. Bảng bank_info gồm (bank_id, bank_name, bank_type)
3. Bảng status_info gồm (trans_status, error_group, message)
4. Bảng customer_profile (customer_id, verified_kyc, dob, name, id_number,
id_address)
*/

-- ví dụ: tạo ra database tên là 'napas'
CREATE DATABASE napas

-- 2. Tạo table:
--- Cách 1: Tạo schema trước
USE napas

DROP TABLE [transaction]

CREATE TABLE [transaction] (
        trans_id INT NOT NULL PRIMARY KEY -- khai báo PK
        , customer_id INT NOT NULL
        , trans_type VARCHAR(50)
        , bank_id INT NOT NULL
        , amount BIGINT
        , trans_status VARCHAR(50)
        , trans_date DATETIME
)

CREATE TABLE [bank_info] (
        bank_id INT NOT NULL PRIMARY KEY
        , bank_name VARCHAR(50)
        , bank_type VARCHAR(50)
)

CREATE TABLE [status_info] (
        trans_status VARCHAR(50) NOT NULL PRIMARY KEY
        , error_group VARCHAR(50)
        , [message] VARCHAR(50)
)
```

```
CREATE TABLE customer_profile (
        customer_id INT NOT NULL PRIMARY KEY
        , verified_kyc VARCHAR(50)
        , dob DATE
        , name VARCHAR(50)
        , id_number INT
        , id_address VARCHAR(50)
        )


SELECT * FROM [transaction]

-- ADD foreign key

ALTER TABLE [transaction]
ADD FOREIGN KEY (customer_id) REFERENCES customer_profile(customer_id)

ALTER TABLE [transaction]
ADD FOREIGN KEY (bank_id) REFERENCES bank_info(bank_id)

ALTER TABLE [transaction]
ADD FOREIGN KEY (trans_status) REFERENCES status_info(trans_status)


-- INSERT dữ liệu

--- cách 1: input bằng tay
--- chèn dữ liệu vào tất cả columns
SELECT * FROM customer_profile
INSERT INTO customer_profile
VALUES (1, 'yes', '1996-01-14', 'Hieu', 12345, 'Q7')
INSERT INTO customer_profile
VALUES (2, 'no', '1995-01-14', 'Duc', 12345, 'Q8')
INSERT INTO customer_profile
VALUES (3, 'no', '1998-01-14', 'Thang', 12345, 'Q9')

SELECT * FROM bank_info
INSERT INTO bank_info
VALUES (1, 'Agribank', 'big4')
INSERT INTO bank_info
```

```
VALUES (2, 'Vietcombank', 'big4')
INSERT INTO bank_info
VALUES (3, 'Techcombank', 'TMCP')

SELECT * FROM status_info
INSERT INTO status_info
VALUES ('success', NULL, NULL)
INSERT INTO status_info
VALUES ('failed', 'bank_error', 'timeout')
INSERT INTO status_info
VALUES (3, 'user_error', NULL)

-- INSERT vào fact : transaction
SELECT * FROM [transaction]
INSERT INTO [transaction]
VALUES (1, 2, 'bike', 2, 10000000, 'success', '2022-10-04')

-- INSERT data từ 1 câu query

SELECT
...
INTO table_name -- cái shema này chưa có
FROM
....

SELECT trans.* , bank_name, bank_type
INTO trans_bank
FROM [transaction] AS trans
JOIN bank_info ON trans.bank_id = bank_info.bank_id

SELECT * FROM trans_bank

-- muốn schema của 1 table

SELECT CONCAT( COLUMN_NAME, ',')
FROM napas.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = N'trans_bank'

-- xóa dữ liệu
-- TRUNCATE: xóa hết các dòng
TRUNCATE TABLE trans_bank

DELETE trans_bank
WHERE ...

SELECT * FROM bank_info

UPDATE bank_info
SET bank_name = 'VPbank'
WHERE bank_name = 'Techcombank'


-- IMPORT FILE CSV vào DATABASE

CREATE DATABASE olist_brazil

SELECT * FROM order_item

-- Thống kê xem state nào đang có nhiều đơn hàng nhất:

SELECT product_id
        , COUNT(order_id) AS number_orders
INTO product_count
FROM order_item
GROUP BY product_id
HAVING COUNT(order_id) > 100
ORDER BY number_orders DESC

SELECT * FROM product_count

--- Tạo VIEW : lưu kết quả xử lý từ truy vấn và tự động update kết quả theo truy vấn
đó
CREATE VIEW name AS
SELECT ....

CREATE VIEW product_count_view AS
SELECT product_id
        , COUNT(order_id) AS number_orders
FROM order_item
```

```
GROUP BY product_id
HAVING COUNT(order_id) > 100
-- ORDER BY number_orders DESC

SELECT * FROM product_count_view
ORDER BY number_orders DESC

DROP VIEW product_count_view


--CORRECT HOMEWORK LESSON 1

/* Task 1: Retrieve data for transportation reports
1.1 Retrieve a list of cities: Initially, you need to produce a list of all of you customers' locations.
Write a Transact-SQL query that queries the SalesLT.Address table and retrieves the values for City and StateProvince
, removing duplicates , then sorts in ascending order of StateProvince and descending order of City. */

-- Thứ tự execution của SQL:
-- FROM -> WHERE -> SELECT (DISTINCT)-> ORDER BY -> LIMIT(TOP N [PERCENT])

SELECT TOP 5 * FROM SalesLT.Address --> nếu mn gặp table mới thì SELECT TOP 5/10 xem format của data

SELECT DISTINCT -- loại bỏ các dòng dữ liệu bị trùng giá trị ở các columns trong lệnh SELECT
	City
	, StateProvince
FROM SalesLT.Address
ORDER BY StateProvince ASC , city DESC

SELECT DISTINCT -- remove duplicates
	StateProvince
	, City
FROM SalesLT.Address
ORDER BY StateProvince ASC , City DESC
```

```
/* 1.2 Retrieve the heaviest products information
Transportation costs are increasing and you need to identify the heaviest products.
Retrieve the names, weight of the top ten percent of products by weight. */


SELECT  * FROM SalesLT.Product -- 295 rows

SELECT TOP 10 PERCENT
        Name
        , Weight
FROM SalesLT.Product
ORDER BY Weight DESC
-- 30 rows




/* Task 2: Retrieve product data
2.1 Filter products by color and size
Retrieve the product number and name of the products
that have a color of black, red, or white and a size of S or M */

SELECT TOP 5 * FROM SalesLT.Product


SELECT
        Name
        , ProductNumber
        , Color
        , Size
FROM
        SalesLT.Product
WHERE
        Color IN ('Black', 'Red', 'White')
        AND Size IN ('S','M')

/*2.2 Filter products by color, size and product number
Retrieve the ProductID, ProductNumber and Name of the products,
- that must have Product number begins with 'BK-'
- followed by any character other than 'T' : kí tự thứ 4 khác T
- and ends with a '-' followed by any two numerals. - và 2 chữ số
```

- And satisfy one of the following conditions: color of black, red, or white, size is S or M and */

```
-- way 1:

-- AND ProductNumber LIKE '%-[a-e][a-e]' -- Kết thúc bởi dấu '-' và 2 chữ cái trong dãy a,b,c,d,e

SELECT ProductID
	, ProductNumber
	, Name
FROM SalesLT.product
WHERE ProductNumber LIKE 'BK-%'
	AND ProductNumber NOT LIKE '___T%' -- kí tự thứ 4 khác chữ T
	AND ProductNumber LIKE '%-[0-9][0-9]' -- Kết thúc bởi dấu '-' và 2 chữ số
	AND ( Color IN ('Black', 'Red', 'White') OR Size IN ('S','M') )

-- 50 rows

-- way 2
SELECT
	ProductID
	, ProductNumber
	, Color
	, Size
	, Name
FROM SalesLT.Product
WHERE
	ProductNumber  LIKE 'BK-[^T]%-[0-9][0-9]'
	AND ( Color IN ('Black', 'Red', 'White')
	OR Size IN ('S','M'))

/*2.3 Retrieve specific products by product ID
Retrieve the product ID, product number, name, and list price of products whose
- product name contains "HL " or "Mountain", --> WHERE Name LIKE
- product number is at least 8 characters --> WHERE ProductNumber
- and never have been ordered. */ -->

SELECT  * FROM SalesLT.Product --> Dimension--> chiều dữ về sản phẩm --> mỗi sản phẩm có 1 dòng
```

--> 295 dòng tức là có tất cả 295 sản phẩm

```
SELECT DISTINCT ProductID FROM SalesLT.SalesOrderDetail --> 142 sản phẩm
đã được bán
--> 153 sản phẩm chưa dc bán

SELECT ProductID
        , name
        , ListPrice
FROM SalesLT.Product
WHERE (Name LIKE '%HL%' OR Name LIKE '%Mountain%')
AND ProductNumber LIKE '________%' -- có ít nhất 8 kí tự
AND ProductID NOT IN ( SELECT DISTINCT ProductID
                FROM SalesLT.SalesOrderDetail) -- tìm những sản phẩm khôn thuộc
danh sách 142 sản phẩm đã bán

-- đáp án: 39 rows --> 39 sản phẩm thỏa yêu cầu



-- CORRECT HOMEWORK LESSON 2
-- Task 1:

/* 1.1 Retrieve customer names and phone numbers
Each customer has an assigned salesperson. You must write a query to create a call
sheet that lists:
   - The salesperson
   - A column named CustomerName that displays how the customer contact should
be greeted
   (for example, Mr Smith)
   - The customer’s phone number. */
-- +
-- CONCAT(col1, col2, col3) ghep cac String va khong bi NULL
SELECT TOP 5 * FROM SalesLT.Customer -- FROM -> WHERE --> SELECT -->
ORDER BY --> LIMIT (TOP)

SELECT
   CustomerID
        , SalesPerson
        , Title
        , Phone
        , ISNULL(Title, ' ') + LastName AS CustomerName
        , CONCAT(Title,' ', LastName) CustomerName_1 -- ignore NULL
        , CONCAT_WS(' ', Title, LastName) AS CustomerName_2 -- ignore NULL
FROM SalesLT.Customer

-- CONCAT: Lệnh dùng để ghép các columns -->
-- CONCAT_WS(special_letters, column1, column2, column3, ..)
--- Syntax: CONCAT(column1, column2, ...)

/* 1.2 Retrieve the heaviest products information
Transportation costs are increasing and you need to identify the heaviest products.
Retrieve the names, weight of the top ten percent of products by weight.
Then, add new column named Number of sell days (caculated from SellStartDate and
SellEndDate)
of these products (if sell end date isn't defined then get Today date)  */
SELECT * FROM SalesLT.Product

-- CASE WHEN
-- IIF

SELECT TOP 10 PERCENT
        ProductID
        , Name
   , Weight
   , SellStartDate
        , SellEndDate
        , CASE
      WHEN SellEndDate IS NULL THEN DATEDIFF(day, SellStartDate,
CURRENT_TIMESTAMP)
        ELSE DATEDIFF(day, SellStartDate, SellEndDate)
        END AS number_of_sell_days
        , DATEDIFF(day, SellStartDate, ISNULL(SellEndDate,
CURRENT_TIMESTAMP)) AS number_of_sell_days_1
        , IIF(SellEndDate IS NULL, DATEDIFF(day, SellStartDate,
CURRENT_TIMESTAMP),
         DATEDIFF(day, SellStartDate, SellEndDate) ) AS number_of_sell_days_2
FROM SalesLT.Product
ORDER BY Weight DESC
```

```
-- total rows: 295 rows --> 10% là ~ 30 rows

-- Task 2:
/* Retrieve a list of customer companies
You have been asked to provide a list of all customer companies in the format
Customer ID : Company Name - for example, 78: Preferred Bikes. */

SELECT * FROM SalesLT.Customer
-- way1: ghép bằng phép +  --> Phải chuyển đổi cho đồng data types (CustomerID -->
nvarchar)
-- way2: Ghép bằng CONCAT --> Khong can quan bi conflict datatype


SELECT
        CustomerID
        , Companyname
        , CAST(CustomerID AS nvarchar) + ': ' + CompanyName AS FormatedName
   , CONCAT(CustomerID, ': ', CompanyName) AS FormatedName_1
FROM SalesLT.Customer


-- 2.2
/* Retrieve a list of sales order revisions
The SalesLT.SalesOrderHeader table contains records of sales orders.
You have been asked to retrieve data for a report that shows:
   - The sales order number and revision number in the format () – for example
SO71774 (2).
   - The order date converted to ANSI standard 102 format (yyyy.mm.dd – for
example 2015.01.31). */

SELECT top 10 * FROM SalesLT.SalesOrderHeader


SELECT
   SalesOrderNumber +' (' + CAST(revisionNumber AS nvarchar) + ')' AS SalesOrder
  , CONVERT(nvarchar,OrderDate, 102) AS OrderDate_ANSI
FROM SalesLT.SalesOrderHeader

--Task 3:
-- 3.1
```

```
/* Retrieve customer contact names with middle names if known
You have been asked to write a query that returns a list of customer names.
The list must consist of a single column in the format first last (for example Keith
Harris)
if the middle name is unknown,
or first middle last (for example Jane M. Gates) if a middle name is known.  */

SELECT * FROM SalesLT.Customer

SELECT
   FirstName
        , MiddleName
        , LastName
        , CONCAT(FirstName,' ', MiddleName, ' ', LastName) AS full_name
        , CONCAT_WS(' ', FirstName, MiddleName, LastName) AS full_name_2 ( là
hàm nâng cao hơn concat khi chèn thêm 1 kí tự gì giữa các chuỗi giống nhau thì nó sẽ
chỉ cần viết 1 lần thôi mà ko cần viết lặp lại các giá trị)
FROM SalesLT.Customer

-- 3.2
/* Retrieve primary contact details
Customers may provide Adventure Works with an email address, a phone number, or
both.
If an email address is available, then it should be used as the primary contact method;
if not, then the phone number should be used. You must write a query that returns a
list of customer IDs in one column,
and a second column named PrimaryContact that contains the email address if known,
and otherwise the phone number. */


SELECT
   CASE WHEN EmailAddress IS NULL THEN Phone
   ELSE EmailAddress
   END pri_contact
   , COALESCE(EmailAddress, Phone) AS pri_contact_1
        , IIF(EmailAddress IS NULL, Phone, EmailAddress) AS pri_contact_2
FROM SalesLT.Customer

-- other ways
SELECT TOP 10
```

```
    CustomerID
    , EmailAddress
          , Phone
    , ISNULL(EmailAddress, Phone) AS PrimaryContact_1
          ,(CASE
       WHEN EmailAddress IS NOT NULL THEN EmailAddress
       ELSE Phone
    END) AS PrimaryContact_2
          , COALESCE(EmailAddress, Phone) AS PrimaryContact_3
FROM SalesLT.Customer

-- TOP: Limit result

SELECT TOP 10 * FROM SalesLT.Customer

--3.3
```

/* As you continue to work with the Adventure Works customer, product and sales data,
you must create queries for reports that have been requested by the sales team.
Retrieve a list of customers with no address
o A sales employee has noticed that Adventure Works does not have address information for all customers.
You must write a query that returns a list of customer IDs, company names,
contact names (first name and last name), and phone numbers for customers with no address stored
in the database. */ (Khi bạn tiếp tục làm việc với khách hàng, dữ liệu sản phẩm và bán hàng của Adventure Works,
Bạn phải tạo truy vấn cho các báo cáo đã được nhóm bán hàng yêu cầu.
Truy xuất danh sách khách hàng không có địa chỉ
o Một nhân viên bán hàng đã nhận thấy rằng Adventure Works không có thông tin địa chỉ cho tất cả khách hàng.
Bạn phải viết một truy vấn trả về danh sách ID khách hàng, tên công ty,
tên liên hệ (họ và tên) và số điện thoại của khách hàng không lưu trữ địa chỉ
trong cơ sở dữ liệu)

```
SELECT * FROM SalesLT.Customer --> 847 rows --> Cty có tổng 847 khách hàng
SELECT * FROM SalesLT.CustomerAddress --> 417 rows --> 407 customer có address --> (may be) 430 customer không có address

Select DISTINCT CustomerID
```

```
from SalesLT.CustomerAddress --> 407 khach hang co Address

--- Your code here
select CustomerID
    , CompanyName
    , FirstName + LastName as Contact_Name
    , Phone
from SalesLT.Customer
where CustomerID NOT IN (Select DISTINCT CustomerID
                    from SalesLT.CustomerAddress) -- 407 rows -- 407 khách hàng có địa
chỉ
-- 440 rows --> 440 khách hàng không có Address

Select DISTINCT CustomerID
from SalesLT.CustomerAddress
-- CORRECT HOMEWORK:

/* Task 1: Generate invoice reports
Adventure Works Cycles sells directly to retailers, who must be invoiced for their
orders.
You have been tasked with writing a query to generate a list of invoices to be sent to
customers.
1.1 Retrieve customer orders
o         As an initial step towards generating the invoice report, write a query that
returns the company name
from the SalesLT.Customer table, and the sales order ID and total due from the
SalesOrderHeader table.
 */

--- Your code here

-- b1: FROM SalesLT.Customer, SalesLT.SalesOrderHeader
-- b2: INNER JOIN 2 table

SELECT TOP 5 * FROM SalesLT.Customer -- dim
SELECT TOP 5 * FROM SalesLT.SalesOrderHeader -- fact

SELECT
          CompanyName
          , SalesOrderID
```

```
        , TotalDue
FROM SalesLT.SalesOrderHeader AS header
JOIN SalesLT.Customer AS cus
        ON header.CustomerID = cus.CustomerID

SELECT  CompanyName
        , SalesOrderID
        , TotalDue
FROM SalesLT.SalesOrderHeader AS header
FULL JOIN SalesLT.Customer AS cus
        ON header.CustomerID = cus.CustomerID
WHERE SalesOrderID IS NOT NULL

-- 32 rows

/* 1.2 Retrieve customer orders with addresses
o       Extend your customer orders query to include the Main Office address for each customer,
including the full street address, city, state or province, postal code, and country or region
o       Tip: Note that each customer can have multiple addressees in the SalesLT.Address table,
so the database developer has created the SalesLT.CustomerAddress table to enable a many-to-many
relationship between customers and addresses. Your query will need to include both of these tables,
and should filter the results so that only Main Office addresses are included.

 */
--- Your code here

SELECT TOP 5 * FROM SalesLT.CustomerAddress
SELECT TOP 5 * FROM SalesLT.Address
SELECT TOP 5 * FROM SalesLT.SalesOrderHeader

SELECT
        cus.CustomerID
        , AddressLine1
        , City
        , StateProvince
        , CountryRegion
        , PostalCode
FROM SalesLT.Customer  AS cus
LEFT JOIN SalesLT.CustomerAddress AS cus_address
        ON cus.CustomerID = cus_address.CustomerID
LEFT JOIN SalesLT.Address AS adress
        ON cus_address.AddressID = adress.AddressID
INNER JOIN SalesLT.SalesOrderHeader AS header -- Lưu ý chỗ này
        ON cus.CustomerID = header.CustomerID
WHERE AddressType = 'Main Office'

-- 857 rows: tất cả khách hàng của cty
-- đáp án thì có 32 rows: vì đây là 32 khách hàng có orders

/* Task 2: Retrieve customer data
As you continue to work with the Adventure Works customer, product and sales data,
you must create queries for reports that have been requested by the sales team.

Retrieve a list of products
○ A sales manager needs a list of ordered product with more information.
You must write a query that returns a
list of product name (is generated by the string preceded by the '-' character (example:
HL Road Frame)),
only started selling in 2006, Product model name contains "Road",
CategoryName contains "Bikes" and  ListPrice value with integer part equal to 2443

*/
SELECT TOP 5 * FROM SalesLT.Product -- có tất cả là 295 sản phẩm
SELECT TOP 5 * FROM SalesLT.ProductModel -- bằng cột ProductModelID
SELECT TOP 5 * FROM SalesLT.ProductCategory -- bằng cột CategoryID
SELECT * FROM SalesLT.SalesOrderDetail -- chứa các sản phẩm đã dc ordered

--- Your code here
-- Xuất phát từ bảng SalesOrderDetail (De lay cac san pham dc ordered)
--> Product --> Model --> Category

SELECT DISTINCT
        detail.ProductID
        , pro.Name AS product_name
        , model.Name AS model_name
        , cat.Name AS cat_name
        , ListPrice
        -- , CHARINDEX('-', pro.Name)
        , CASE
        WHEN CHARINDEX('-', pro.Name) = 0 THEN pro.Name
        ELSE SUBSTRING(pro.Name, 1, CHARINDEX('-', pro.Name) -1)
        END AS modified_name_1
        , LEFT (pro.Name, IIF( CHARINDEX('-', pro.Name) = 0, LEN(pro.Name),
CHARINDEX('-', pro.Name) -1 )) AS modified_name_2
FROM SalesLT.SalesOrderDetail AS detail -- Chú ý chỗ này
INNER JOIN SalesLT.Product  AS pro
        ON detail.ProductID = pro.ProductID
INNER JOIN SalesLT.ProductModel AS model
        ON pro.ProductModelID = model.ProductModelID
INNER JOIN SalesLT.ProductCategory AS cat
        ON pro.ProductCategoryID = cat.ProductCategoryID
WHERE YEAR(SellStartDate) = 2006 -- SellStartDate BETWEEN '2006-01-01'
AND '2006-12-31'
        AND model.Name LIKE '%Road%'
        AND cat.Name LIKE '%Bikes%'
        AND CAST(ListPrice AS INT) = 2443
        -- AND FLOOR (ListPrice) = 2443 -- LIKE '2443%'

-- đáp án: 5 rows

-- PART 2: payTM

/* Task 1: Retrieve reports on transaction scenarios
1.1 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, transaction_type, sub_category, category.
These transactions must meet the following conditions:
•               Were created in Jan 2019
•               Transaction type is not payment */

-- Your code here

SELECT TOP 5 * FROM fact_transaction_2019 -- fact
SELECT TOP 5 * FROM dim_scenario -- dim

SELECT
```

```
        customer_id
        , transaction_id
        , fact.scenario_id
        , transaction_type
        , sub_category
        , category
        , transaction_time
FROM fact_transaction_2019 AS fact
LEFT JOIN dim_scenario AS scena
        ON fact.scenario_id = scena.scenario_id
WHERE transaction_time BETWEEN '2019-01-01' AND '2019-02-01'
        --MONTH(transaction_time) = 1 -- transaction_time < '2019-02-01'
        AND transaction_type != 'Payment'
ORDER BY transaction_time DESC

-- khác biệt between giữa datetime và int

-- 7619 rows

SELECT DISTINCT transaction_type
FROM dim_scenario

-- 7619 rows

/* 1.2 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, transaction_type, category,
payment_method.
These transactions must meet the following conditions:
•               Were created from Jan to June 2019
•               Had category type is shopping
•               Were paid by Bank account
*/
-- Your code here
SELECT TOP 5 * FROM fact_transaction_2019
SELECT TOP 5 * FROM dim_scenario
SELECT TOP 5 * FROM dim_payment_channel

SELECT
        customer_id
        , transaction_id
        , fact.scenario_id
        , transaction_type
        , category
        , payment_method
FROM fact_transaction_2019 AS fact
LEFT JOIN dim_scenario AS scena
        ON fact.scenario_id = scena.scenario_id
LEFT JOIN dim_payment_channel AS channel
        ON fact.payment_channel_id = channel.payment_channel_id
WHERE transaction_time < '2019-07-01' -- MONTH(transaction_time) <= 6
        AND category = 'Shopping'
        AND payment_method = 'Bank account'

category LIKE 'Shopping' -- LIKE sẽ chạy lâu hơn với phép =
MONTH(transaction_time) IN (1,2,3,4,5,6) -- cách IN nhiều giá trị nó sẽ xử lý lâu
hơn

-- 600 rows

/* 1.3 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, payment_method and payment_platform.
These transactions must meet the following conditions:
•               Were created in Jan 2019 and Jan 2020
•               Had payment platform is android
*/
-- Your code here
SELECT TOP 5 * FROM dim_platform
SELECT TOP 5 * FROM fact_transaction_2019 -- 300K rows
SELECT TOP 5 * FROM fact_transaction_2020 -- 700K rows

-- way 1: UNION trước 2 fact tables sau đó mới đi JOIN và đặt điều kiện
fact_19 UNION fact_20 - JOIN dim_platform

-- way 2: JOIN từng table fact với dim sau đó UNION lại --> performance tốt hơn
cách 1
fact_19 JOIN dim_platform , đặt các điều ios
fact_20 JOIN dim_platform , đặt các điều ios
UNION 2 kết quả trên

-- way1 : Union xong rồi mới JOIN
```

```
SELECT *
FROM
        ( SELECT *
        FROM fact_transaction_2019
        UNION
        SELECT *
        FROM fact_transaction_2020 ) AS fact_table -- > 1 triệu dòng thì sẽ nhiều và
chạy lâu hơn
LEFT JOIN dim_platform AS platform
        ON fact_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_table.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH (transaction_time) = 1

-- 35,297, 2s

-- way 2:
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2019 AS fact_19
JOIN dim_platform AS plat
ON fact_19.platform_id = plat.platform_id
JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH(transaction_time) = 1 -- 9,929
rows thỏa mãn trong 2019
UNION -- gop lai bang UNION
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2020 fact_20
JOIN dim_platform plat
ON fact_20.platform_id = plat.platform_id
JOIN dim_payment_channel channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH(transaction_time) = 1 --
25,368 rows thỏa mãn trong 2020

-- 35,297 rows , 2s

-- FROM-> JOIN -> WHERE -> SELECT -> UNION
-- cách viết sai
SELECT transaction_id
FROM fact_transaction_2019 fact_19 -- 396K
UNION
SELECT transaction_id
FROM fact_transaction_2020 fact_20
JOIN dim_platform plat
ON fact_20.platform_id = plat.platform_id
JOIN dim_payment_channel AS channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'ios' AND MONTH(transaction_time) = 1 -- 31,955

-- 428,772



/* 1.4 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, payment_method and payment_platform.
These transactions must meet the following conditions:
•               Include all transactions of the customer group created in January 2019
(Group A)
and additional transactions of this customers (Group A) continue to make transactions
in January 2020.
•               Payment platform is iOS */

-- ví dụ tháng 1/2019 có 1000 customers --> lấy hết giao dịch (1/2019)
-- Đi tìm thêm các giao dịch của 1000 customers trên phát sinh trong tháng 1/2020

-- Your code here:
-- Đi tìm danh sách khách hàng trong tháng 1 2019:
SELECT DISTINCT customer_id
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_platform AS platform
        ON fact_19.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios' -- group A có 3,419 customers
```

```
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_platform AS platform
        ON fact_19.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios' -- 11,783 rows của Group A phát sinh trong
tháng 1/2019
UNION
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform -- 9,007 rows
FROM fact_transaction_2020 AS fact_20
LEFT JOIN dim_platform AS platform
        ON fact_20.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios'
        AND customer_id IN ( SELECT DISTINCT customer_id
                FROM fact_transaction_2019 AS fact_19
                LEFT JOIN dim_platform AS platform
                ON fact_19.platform_id = platform.platform_id
                LEFT JOIN dim_payment_channel AS channel
                ON fact_19.payment_channel_id = channel.payment_channel_id
                WHERE MONTH(transaction_time) = 1
                AND payment_platform = 'ios'
                ) -- 8,179 giao dịch của group A phát sinh giao dịch trong 1/2020
-- đáp án: 19,962 rows
```

```sql
-- CORRECT HOMEWORK:

/* Task 1: Generate invoice reports
Adventure Works Cycles sells directly to retailers, who must be invoiced for their
orders.
You have been tasked with writing a query to generate a list of invoices to be sent to
customers.
1.1 Retrieve customer orders
o        As an initial step towards generating the invoice report, write a query that
returns the company name
from the SalesLT.Customer table, and the sales order ID and total due from the
SalesOrderHeader table.
 */

--- Your code here

-- b1: FROM SalesLT.Customer, SalesLT.SalesOrderHeader
-- b2: INNER JOIN 2 table

SELECT TOP 5 * FROM SalesLT.Customer -- dim
SELECT TOP 5 * FROM SalesLT.SalesOrderHeader -- fact

SELECT
        CompanyName
        , SalesOrderID
        , TotalDue
FROM SalesLT.SalesOrderHeader AS header
JOIN SalesLT.Customer AS cus
        ON header.CustomerID = cus.CustomerID

SELECT  CompanyName
        , SalesOrderID
        , TotalDue
FROM SalesLT.SalesOrderHeader AS header
FULL JOIN SalesLT.Customer AS cus
        ON header.CustomerID = cus.CustomerID
```

```
WHERE SalesOrderID IS NOT NULL

-- 32 rows

/* 1.2 Retrieve customer orders with addresses
o	Extend your customer orders query to include the Main Office address for each customer,
including the full street address, city, state or province, postal code, and country or region
o	Tip: Note that each customer can have multiple addressees in the SalesLT.Address table,
so the database developer has created the SalesLT.CustomerAddress table to enable a many-to-many
relationship between customers and addresses. Your query will need to include both of these tables,
and should filter the results so that only Main Office addresses are included.

 */
--- Your code here

SELECT TOP 5 * FROM SalesLT.CustomerAddress
SELECT TOP 5 * FROM SalesLT.Address
SELECT TOP 5 * FROM SalesLT.SalesOrderHeader

SELECT
	cus.CustomerID
	, AddressLine1
	, City
	, StateProvince
	, CountryRegion
	, PostalCode
FROM SalesLT.Customer  AS cus
LEFT JOIN SalesLT.CustomerAddress AS cus_address
	ON cus.CustomerID = cus_address.CustomerID
LEFT JOIN SalesLT.Address AS adress
	ON cus_address.AddressID = adress.AddressID
INNER JOIN SalesLT.SalesOrderHeader AS header -- Lưu ý chỗ này
	ON cus.CustomerID = header.CustomerID
WHERE AddressType = 'Main Office'
```

```
-- 857 rows: tất cả khách hàng của cty
-- đáp án thì có 32 rows: vì đây là 32 khách hàng có orders

/* Task 2: Retrieve customer data
As you continue to work with the Adventure Works customer, product and sales data,
you must create queries for reports that have been requested by the sales team.

Retrieve a list of products
○ A sales manager needs a list of ordered product with more information.
You must write a query that returns a
list of product name (is generated by the string preceded by the '-' character (example:
HL Road Frame)),
only started selling in 2006, Product model name contains "Road",
CategoryName contains "Bikes" and  ListPrice value with integer part equal to 2443

*/
SELECT TOP 5 * FROM SalesLT.Product -- có tất cả là 295 sản phẩm
SELECT TOP 5 * FROM SalesLT.ProductModel -- bằng cột ProductModelID
SELECT TOP 5 * FROM SalesLT.ProductCategory -- bằng cột CategoryID
SELECT * FROM SalesLT.SalesOrderDetail -- chứa các sản phẩm đã dc ordered

--- Your code here
-- Xuất phát từ bảng SalesOrderDetail (De lay cac san pham dc ordered)
--> Product --> Model --> Category

SELECT DISTINCT
	detail.ProductID
	, pro.Name AS product_name
	, model.Name AS model_name
	, cat.Name AS cat_name
	, ListPrice
	-- , CHARINDEX('-', pro.Name)
	, CASE
	WHEN CHARINDEX('-', pro.Name) = 0 THEN pro.Name
	ELSE SUBSTRING(pro.Name, 1, CHARINDEX('-', pro.Name) -1)
	END AS modified_name_1
	, LEFT (pro.Name, IIF( CHARINDEX('-', pro.Name) = 0, LEN(pro.Name),
CHARINDEX('-', pro.Name) -1 )) AS modified_name_2
FROM SalesLT.SalesOrderDetail AS detail -- Chú ý chỗ này
INNER JOIN SalesLT.Product  AS pro
```

```
        ON detail.ProductID = pro.ProductID
INNER JOIN SalesLT.ProductModel AS model
        ON pro.ProductModelID = model.ProductModelID
INNER JOIN SalesLT.ProductCategory AS cat
        ON pro.ProductCategoryID = cat.ProductCategoryID
WHERE YEAR(SellStartDate) = 2006 -- SellStartDate BETWEEN '2006-01-01'
AND '2006-12-31'
        AND model.Name LIKE '%Road%'
        AND cat.Name LIKE '%Bikes%'
        AND CAST(ListPrice AS INT) = 2443
        -- AND FLOOR (ListPrice) = 2443 -- LIKE '2443%'

-- đáp án: 5 rows

-- PART 2: payTM

/* Task 1: Retrieve reports on transaction scenarios
1.1 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, transaction_type, sub_category, category.
These transactions must meet the following conditions:
•       Were created in Jan 2019
•       Transaction type is not payment */

-- Your code here

SELECT TOP 5 * FROM fact_transaction_2019 -- fact
SELECT TOP 5 * FROM dim_scenario -- dim

SELECT
        customer_id
        , transaction_id
        , fact.scenario_id
        , transaction_type
        , sub_category
        , category
        , transaction_time
FROM fact_transaction_2019 AS fact
LEFT JOIN dim_scenario AS scena
        ON fact.scenario_id = scena.scenario_id
WHERE transaction_time BETWEEN '2019-01-01' AND '2019-02-01'
```

```
        --MONTH(transaction_time) = 1 -- transaction_time < '2019-02-01'
        AND transaction_type != 'Payment'
ORDER BY transaction_time DESC

-- khác biệt between giữa datetime và int

-- 7619 rows

SELECT DISTINCT transaction_type
FROM dim_scenario

-- 7619 rows

/* 1.2 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, transaction_type, category,
payment_method.
These transactions must meet the following conditions:
•               Were created from Jan to June 2019
•               Had category type is shopping
•               Were paid by Bank account
*/
-- Your code here
SELECT TOP 5 * FROM fact_transaction_2019
SELECT TOP 5 * FROM dim_scenario
SELECT TOP 5 * FROM dim_payment_channel

SELECT
        customer_id
        , transaction_id
        , fact.scenario_id
        , transaction_type
        , category
        , payment_method
FROM fact_transaction_2019 AS fact
LEFT JOIN dim_scenario AS scena
        ON fact.scenario_id = scena.scenario_id
LEFT JOIN dim_payment_channel AS channel
        ON fact.payment_channel_id = channel.payment_channel_id
WHERE transaction_time < '2019-07-01' -- MONTH(transaction_time) <= 6
        AND category = 'Shopping'
```

```
	AND payment_method = 'Bank account'

category LIKE 'Shopping' -- LIKE sẽ chạy lâu hơn với phép =
MONTH(transaction_time) IN (1,2,3,4,5,6) -- cách IN nhiều giá trị nó sẽ xử lý lâu
hơn

-- 600 rows

/* 1.3 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, payment_method and payment_platform.
These transactions must meet the following conditions:
•		Were created in Jan 2019 and Jan 2020
•		Had payment platform is android
*/
-- Your code here
SELECT TOP 5 * FROM dim_platform
SELECT TOP 5 * FROM fact_transaction_2019 -- 300K rows
SELECT TOP 5 * FROM fact_transaction_2020 -- 700K rows

-- way 1: UNION trước 2 fact tables sau đó mới đi JOIN và đặt điều kiện
fact_19 UNION fact_20 - JOIN dim_platform

-- way 2: JOIN từng table fact với dim sau đó UNION lại --> performance tốt hơn
cách 1
fact_19 JOIN dim_platform , đặt các điều ios
fact_20 JOIN dim_platform , đặt các điều ios
UNION 2 kết quả trên

-- way1 : Union xong rồi mới JOIN
SELECT *
FROM
	( SELECT *
	FROM fact_transaction_2019
	UNION
	SELECT *
	FROM fact_transaction_2020 ) AS fact_table -- > 1 triệu dòng thì sẽ nhiều và
chạy lâu hơn
LEFT JOIN dim_platform AS platform
	ON fact_table.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
```

```
        ON fact_table.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH (transaction_time) = 1

-- 35,297, 2s

-- way 2:
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2019 AS fact_19
JOIN dim_platform AS plat
ON fact_19.platform_id = plat.platform_id
JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH(transaction_time) = 1 -- 9,929
rows thỏa mãn trong 2019
UNION -- gop lai bang UNION
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2020 fact_20
JOIN dim_platform plat
ON fact_20.platform_id = plat.platform_id
JOIN dim_payment_channel channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE payment_platform = 'android' AND MONTH(transaction_time) = 1 --
25,368 rows thỏa mãn trong 2020

-- 35,297 rows , 2s

-- FROM-> JOIN -> WHERE -> SELECT -> UNION

-- cách viết sai
SELECT transaction_id
FROM fact_transaction_2019 fact_19 -- 396K
UNION
SELECT transaction_id
FROM fact_transaction_2020 fact_20
JOIN dim_platform plat
ON fact_20.platform_id = plat.platform_id
JOIN dim_payment_channel AS channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
```

```
WHERE payment_platform = 'ios' AND MONTH(transaction_time) = 1 -- 31,955

-- 428,772



/* 1.4 Retrieve a report that includes the following information:
customer_id, transaction_id, scenario_id, payment_method and payment_platform.
These transactions must meet the following conditions:
•               Include all transactions of the customer group created in January 2019
(Group A)
and additional transactions of this customers (Group A) continue to make transactions
in January 2020.
•               Payment platform is iOS */

-- ví dụ tháng 1/2019 có 1000 customers --> lấy hết giao dịch (1/2019)
-- Đi tìm thêm các giao dịch của 1000 customers trên phát sinh trong tháng 1/2020

-- Your code here:
-- Đi tìm danh sách khách hàng trong tháng 1 2019:
SELECT DISTINCT customer_id
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_platform AS platform
        ON fact_19.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios' -- group A có 3,419 customers


SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_platform AS platform
        ON fact_19.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_19.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios' -- 11,783 rows của Group A phát sinh trong
tháng 1/2019
```

```
UNION
SELECT customer_id, transaction_id, scenario_id, payment_method,
payment_platform -- 9,007 rows
FROM fact_transaction_2020 AS fact_20
LEFT JOIN dim_platform AS platform
        ON fact_20.platform_id = platform.platform_id
LEFT JOIN dim_payment_channel AS channel
        ON fact_20.payment_channel_id = channel.payment_channel_id
WHERE MONTH(transaction_time) = 1
        AND payment_platform = 'ios'
        AND customer_id IN ( SELECT DISTINCT customer_id
                FROM fact_transaction_2019 AS fact_19
                LEFT JOIN dim_platform AS platform
                ON fact_19.platform_id = platform.platform_id
                LEFT JOIN dim_payment_channel AS channel
                ON fact_19.payment_channel_id = channel.payment_channel_id
                WHERE MONTH(transaction_time) = 1
                AND payment_platform = 'ios'
                ) -- 8,179 giao dịch của group A phát sinh giao dịch trong 1/2020
-- đáp án: 19,962 rows
```

-- CORRECT HOMEWORK 4: SUBQUERY - GROUP BY - CTE

/* Task 1: Retrieve an overview report of payment types
1.1. Paytm has a wide variety of transaction types in its business.
Your manager wants to know the contribution (by percentage) of each transaction type to total transactions.
Retrieve a report that includes the following information: transaction type,
number of transaction and proportion of each type in total.
These transactions must meet the following conditions:

- Were created in 2019
- Were paid successfully

Show only the results of the top 5 types with the highest percentage of the total. */
(Truy xuất báo cáo tổng quan về các loại thanh toán
1.1. Paytm có nhiều loại giao dịch khác nhau trong hoạt động kinh doanh của mình.
Người quản lý của bạn muốn biết sự đóng góp (theo tỷ lệ phần trăm) của từng loại giao dịch vào tổng số giao dịch.
Truy xuất báo cáo bao gồm các thông tin sau: loại giao dịch,
số lượng giao dịch và tỷ lệ của từng loại trong tổng số.

Các giao dịch này phải đáp ứng các điều kiện sau:
• Được tạo vào năm 2019
• Đã thanh toán thành công
Chỉ hiển thị kết quả của 5 loại hàng đầu với tỷ lệ phần trăm cao nhất trong tổng số. */)

```
-- Your code here
-- b1: JOIN 3 tables: fact_transaction_2019, dim_scenario, status --> LEFT JOIN từ
fact và lấy success
-- b2: Gom nhóm theo transaction type -> tính số giao dịch --> GROUP BY , COUNT
(transaction_id)
-- b3: Tính tổng số giao dịch success của 2019 --> SUBQUERY để đếm số giao dịch
2019
-- b4: Tỉnh tỉ trọng = b1/b2
-- b5: Chọn top 5 cao nhất --> SELECT TOP 5 , ORDER BY ...

WITH joined_table AS ( -- b1
SELECT fact_19.*, transaction_type
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS stat
        ON fact_19.status_id = stat.status_id
WHERE status_description = 'success'
)
, total_table AS (
SELECT transaction_type -- group by cái gì select cái đó
        , COUNT(transaction_id) AS number_trans
        , (SELECT COUNT(transaction_id) FROM joined_table) AS total_trans
FROM joined_table
GROUP BY transaction_type
)
SELECT TOP 5
        *
        , FORMAT ( number_trans*1.0/total_trans, 'p') AS pct  --> SQL trả ra INT,
0.4732
FROM total_table
ORDER BY number_trans DESC
```

/* 1.2. After your manager looks at the results of these top 5 types,

he wants to deep dive more to gain more insights.
Retrieve a more detailed report with following information: transaction type, category, number of transaction and proportion of each category in the total of that transaction type.
These transactions must meet the following conditions:
- Were created in 2019
- Were paid successfully */

```
-- Your code here
-- b1: JOIN facc19 , scenario, status
-- b2: Group by theo type, category để tìm mỗi category có bao nhiêu trans
-- b3: Group by theo type để tìm mỗi type có bao nhiêu trans
-- b4: JOIN 2 kết quả trên lại
-- b5: tính pct

WITH join_table AS ( -- b1
SELECT fact_19.*, transaction_type, category
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS stat
        ON fact_19.status_id = stat.status_id
WHERE status_description = 'success'
)
, count_category AS ( -- b2
SELECT transaction_type, category
        , COUNT(transaction_id) AS number_trans_category
FROM join_table
GROUP BY transaction_type, category
)
, count_type AS ( -- b3
SELECT transaction_type
        , COUNT(transaction_id) AS number_trans_type
FROM join_table
GROUP BY transaction_type
)
SELECT count_category.*, number_trans_type -- b4
        , FORMAT( number_trans_category*1.0/number_trans_type, 'p') AS pct
FROM count_category
FULL JOIN count_type
```

```
ON count_category.transaction_type = count_type.transaction_type
WHERE number_trans_type IS NOT NULL AND number_trans_category IS NOT
NULL
ORDER BY number_trans_category*1.0/number_trans_type DESC


/* Task 2: Retrieve an overview report of customer's payment behaviors
2.1. Paytm has acquired a lot of customers.
Retrieve a report that includes the following information: the number of transactions,
the number of payment scenarios, the number of transaction types,
the number of payment category and the total of charged amount of each customer.

•		Were created in 2019
•		Had status description is successful
•		Had transaction type is payment
•		Only show Top 10 highest customers by the number of transactions */

-- Your code here
-- b1: Join tables
-- b2: Đặt điều kiện status và type
-- b3: group by customer_id --> COUNT và SUM để tính các chỉ số

SELECT
	-- TOP 10
	customer_id
	, COUNT(transaction_id) AS number_trans
	, COUNT(DISTINCT fact_19.scenario_id) AS number_scenarios
	, COUNT(DISTINCT scena.category) AS number_categories
	, SUM(charged_amount*1.0) AS total_amount
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
	ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS sta
	ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
	AND transaction_type = 'payment'
GROUP BY customer_id
ORDER BY number_trans DESC

/* 2.2. After looking at the above metrics of customer's payment behaviors,
```

we want to analyze the distribution of each metric. Before calculating and plotting the distribution
to check the frequency of values in each metric, we need to group the observations into range.
2.2.1. How can we group the observations in the most logical way? Binning is useful to help us deal with problem. To use binning method, we need to determine how many bins for
each distribution of each field.
Retrieve a report that includes the following columns: metric, minimum value, maximum value
and average value of these metrics:

- The total charged amount
- The number of transactions
- The number of payment scenarios
- The number of payment categories */

```
-- The number of transactions

WITH summary_table AS (
SELECT customer_id
        , COUNT(transaction_id) AS number_trans
        , COUNT(DISTINCT fact_19.scenario_id) AS number_scenarios
        , COUNT(DISTINCT scena.category) AS number_categories
        , SUM(charged_amount) AS total_amount
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS sta
        ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
        AND transaction_type = 'payment'
GROUP BY customer_id
)
SELECT 'The number of transaction' AS metric
        , MIN(number_trans) AS min_value
        , MAX(number_trans) AS max_value
        , AVG(number_trans) AS avg_value
FROM summary_table
UNION
```

```
SELECT 'The number of scenarios' AS metric
        , MIN(number_scenarios) AS min_value
        , MAX(number_scenarios) AS max_value
        , AVG(number_scenarios) AS avg_value
FROM summary_table
UNION
SELECT 'The number of categories' AS metric
        , MIN(number_categories) AS min_value
        , MAX(number_categories) AS max_value
        , AVG(number_categories) AS avg_value
FROM summary_table
UNION
SELECT 'The total charged amount' AS metric
        , MIN(total_amount) AS min_value
        , MAX(total_amount) AS max_value
        , AVG(1.0*total_amount) AS avg_value
FROM summary_table


/* Bin the total charged amount and number of transactions then calculate the
frequency of each field in each metric

Metric 3: The total charged amount */

WITH summary_table AS (
SELECT customer_id
        , SUM(charged_amount) AS total_amount
        , CASE
        WHEN SUM(charged_amount) < 1000000 THEN '0-01M'
        WHEN SUM(charged_amount) >= 1000000 AND SUM(charged_amount) <
2000000 THEN '01M-02M'
        WHEN SUM(charged_amount) >= 2000000 AND SUM(charged_amount) <
3000000 THEN '02M-03M'
        WHEN SUM(charged_amount) >= 3000000 AND SUM(charged_amount) <
4000000 THEN '03M-04M'
        WHEN SUM(charged_amount) >= 4000000 AND SUM(charged_amount) <
5000000 THEN '04M-05M'
        WHEN SUM(charged_amount) >= 5000000 AND SUM(charged_amount) <
6000000 THEN '05M-06M'
```

```
        WHEN SUM(charged_amount) >= 6000000 AND SUM(charged_amount) <
7000000 THEN '06M-07M'
        WHEN SUM(charged_amount) >= 7000000 AND SUM(charged_amount) <
8000000 THEN '07M-08M'
        WHEN SUM(charged_amount) >= 8000000 AND SUM(charged_amount) <
9000000 THEN '08M-09M'
        WHEN SUM(charged_amount) >= 9000000 AND SUM(charged_amount) <
10000000 THEN '09M-10M'
        WHEN SUM(charged_amount) >= 10000000 THEN 'more > 10M'
        END AS charged_amount_range
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS sta
        ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
        AND transaction_type = 'payment'
GROUP BY customer_id
)
SELECT charged_amount_range
        , COUNT(customer_id) AS number_customers
FROM summary_table
GROUP BY charged_amount_range
ORDER BY charged_amount_range

-- Metric 1: The number of payment categories */
WITH summary_table AS (
SELECT customer_id
        , COUNT(DISTINCT scena.category) AS number_categories
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS sta
        ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
        AND transaction_type = 'payment'
GROUP BY customer_id
)
SELECT number_categories
        , COUNT(customer_id) AS number_customers
```

```
FROM summary_table
GROUP BY number_categories
ORDER BY number_categories

-- Metric 2: The number of payment scenarios
WITH summary_table AS (
SELECT customer_id
        , COUNT(DISTINCT fact_19.scenario_id) AS number_scenarios
FROM fact_transaction_2019 AS fact_19
LEFT JOIN dim_scenario AS scena
        ON fact_19.scenario_id = scena.scenario_id
LEFT JOIN dim_status AS sta
        ON fact_19.status_id = sta.status_id
WHERE status_description = 'success'
        AND transaction_type = 'payment'
GROUP BY customer_id
)
SELECT number_scenarios
        , COUNT(customer_id) AS number_customers
FROM summary_table
GROUP BY number_scenarios
ORDER BY number_scenarios

-- CORRECT HOMEWORK 6 + Lesson 7: Time Series Analysis

/* 1.1. Simple trend
Task: You need to analyze the trend of payment transactions of Billing category from
2019 to 2020.
First, let’s show the trend of the number of successful transaction by month. */
-- các loại hóa đơn:

-- b1: data source fact 19 và fact 20 , dim scenario
-- b2:
--- cách 1: Gộp 2 bảng 19 và 20 lại --> toàn bộ fact transaction --> JOIN để tìm
Billing: UNION fact 19 và fact 20 ; LEFT JOIN dim scenario
--- cách 2: JOIN lần lượt từng bảng fact với scenario --> UNION 2 data tables lại  |
LEFT JOIN từ fact sang dim và UNION sau
-- b3: gom nhóm theo tháng và đếm số giao dịch --> GROUP BY month và
COUNT(transaction_id)
```

```
-- Đáp án

-- cách 1: UNION 2 bảng trước --> JOIN --> gom nhóm tính toán

WITH fact_table AS ( -- 1,198,484 rows
SELECT transaction_id, transaction_time, status_id, scenario_id
FROM fact_transaction_2019 -- 396k rows
UNION
SELECT transaction_id, transaction_time, status_id, scenario_id
FROM fact_transaction_2020 )  -- 700k rows)
SELECT
        Year(transaction_time) AS year, Month(transaction_time) AS month
        , CONVERT(nvarchar(6), transaction_time, 112) AS time_calendar
        , COUNT(transaction_id) AS number_trans
FROM fact_table
JOIN dim_scenario AS sce ON fact_table.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
GROUP BY Year(transaction_time), Month(transaction_time),
CONVERT(nvarchar(6), transaction_time, 112)
ORDER BY year, month
-- 4s

-- cách 2: JOIN từng bảng FACT với Scenario và đặt điều kiện Billing --> UNION
WITH fact_table AS (
SELECT fact_19.*, category
FROM fact_transaction_2019 fact_19
JOIN dim_scenario sce
ON fact_19.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
UNION
SELECT fact_20.*, category
FROM fact_transaction_2020 fact_20
JOIN dim_scenario sce
ON fact_20.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
)
SELECT      Year(transaction_time) AS year, Month(transaction_time) AS month
        , CONVERT(nvarchar(6), transaction_time, 112) AS time_calendar
        , COUNT(transaction_id) AS number_trans
FROM fact_table
```

```
GROUP BY Year(transaction_time), Month(transaction_time),
CONVERT(nvarchar(6), transaction_time, 112)
ORDER BY year, month
-- 3s:

-- 1.2

WITH fact_table AS (
SELECT *
FROM fact_transaction_2019
UNION
SELECT *
FROM fact_transaction_2020 )
SELECT
        YEAR(transaction_time) AS year, MONTH(transaction_time) AS month
        , sub_category
        , COUNT(transaction_id) AS number_trans
FROM fact_table
JOIN dim_scenario AS sce ON fact_table.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
GROUP BY YEAR(transaction_time), MONTH(transaction_time), sub_category
ORDER BY year, month

-- COUNT(transaction_id) OVER ( PARTITION BY month, sub_category) AS
number_trans

-- Modifying kết quả (PIVOT TABLE)

-- cách 1: pivot bằng cách group by và aggregate có case when --> MS SQL Server ,
Postgres SQL, MySQL (Ưu tiên cách này, dùng ở đâu cũng dc)

WITH fact_table AS (
SELECT *
FROM fact_transaction_2019
UNION
SELECT *
FROM fact_transaction_2020 )
, sub_month AS (
SELECT
        YEAR(transaction_time) AS year, MONTH(transaction_time) AS month
        , sub_category
        , COUNT(transaction_id) AS number_trans
FROM fact_table
JOIN dim_scenario AS sce ON fact_table.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
GROUP BY YEAR(transaction_time), MONTH(transaction_time), sub_category
)
SELECT year, month
        , SUM ( CASE WHEN sub_category = 'Electricity' THEN number_trans END
) AS elec_trans
        , SUM ( CASE WHEN sub_category = 'Internet' THEN number_trans END )
AS internet_trans
        , SUM ( CASE WHEN sub_category = 'Water' THEN number_trans END ) AS
water_trans
FROM sub_month
GROUP BY year, month
ORDER BY year, month
```

-- cách 2: dùng hàm PIVOT của MS SQL Server

Cú pháp :

```
SELECT ...
FROM
PIVOT (
        Aggregate function
        FOR column_pivot IN ("Electricity", "Internet", "Water")
)
```

```
WITH fact_table AS (
SELECT *
FROM fact_transaction_2019
UNION
SELECT *
FROM fact_transaction_2020 )
, sub_month AS (
SELECT
        YEAR(transaction_time) AS year, MONTH(transaction_time) AS month
        , sub_category
        , COUNT(transaction_id) AS number_trans
FROM fact_table
JOIN dim_scenario AS sce ON fact_table.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
GROUP BY YEAR(transaction_time), MONTH(transaction_time), sub_category
)
SELECT year, month -- non-pivot columns
        , "Electricity" AS elec_trans
        , "Internet" AS inter_trans
        , "Water" AS water_trans
FROM (
        SELECT year, month, sub_category, number_trans
        FROM sub_month
) AS source_table
PIVOT (
        SUM(number_trans) -- aggregate funtion
        FOR sub_category IN ( "Electricity", "Internet", "Water" ) -- khai báo column
muốn pivot, cụ thể là muốn pivot giá trị nào
) AS pivot_table
ORDER BY year, month


-- 1.3 Percent of total
WITH fact_table AS (
SELECT *
FROM fact_transaction_2019
UNION
SELECT *
FROM fact_transaction_2020 )
, sub_count AS (
SELECT
        YEAR(transaction_time) year, MONTH(transaction_time) month
        , sub_category
        , COUNT(transaction_id) AS number_trans
FROM fact_table
JOIN dim_scenario AS sce ON fact_table.scenario_id = sce.scenario_id
WHERE status_id = 1 AND category = 'Billing'
GROUP BY YEAR(transaction_time), MONTH(transaction_time), sub_category
)
, sub_month AS (
SELECT Year
        , month
        , SUM( CASE WHEN sub_category = 'Electricity' THEN number_trans ELSE
0 END ) AS electricity_trans
        , SUM( CASE WHEN sub_category = 'Internet' THEN number_trans ELSE 0
END ) AS internet_trans
        , SUM( CASE WHEN sub_category = 'Water' THEN number_trans ELSE 0
END ) AS water_trans
FROM sub_count
GROUP BY year, month
)
, total_month AS (
        SELECT *
        , ISNULL(electricity_trans,0) + ISNULL(internet_trans,0) +
ISNULL(water_trans,0) AS total_trans_month
FROM sub_month
)
SELECT *
        , FORMAT(1.0*electricity_trans/total_trans_month, 'p') AS elec_pct
        , FORMAT(1.0*internet_trans/total_trans_month, 'p') AS iternet_pct
        , FORMAT(1.0*water_trans/total_trans_month, 'p') AS water_pct
FROM total_month

-- 1.4

WITH fact_table AS (
SELECT * FROM fact_transaction_2019
UNION
SELECT * FROM fact_transaction_2020
)
, customer_month AS (
SELECT MONTH(transaction_time) month, YEAR(transaction_time) year
        , COUNT( DISTINCT customer_id ) AS number_customer -- đếm số lượng
khách hàng
FROM fact_table
JOIN dim_scenario AS scena ON fact_table.scenario_id = scena.scenario_id
WHERE category = 'Billing' AND status_id = 1 AND sub_category IN ('Electricity',
'Internet',  'Water')
GROUP BY MONTH(transaction_time), YEAR(transaction_time)
)
```

```
SELECT *
	, start_point = (SELECT number_customer FROM customer_month WHERE
year = 2019 AND month = 1)
	, start_point_1 = FIRST_VALUE(number_customer) OVER (ORDER BY
year, month)
	, FORMAT (1.0*number_customer/FIRST_VALUE(number_customer) OVER
(ORDER BY year, month) -1 , 'p') AS diff_pct
FROM customer_month

-- 2. Rolling time window

/* 2.1 Task: Select only these sub-categories in the list (Electricity, Internet and
Water),
you need to calculate the number of successful paying customers for each week
number from 2019 to 2020).
Then get rolling annual paying users of total. */

select datepart(week, '2022-09-27');

WITH fact_table AS (
SELECT * FROM fact_transaction_2019
UNION
SELECT * FROM fact_transaction_2020
)
, week_user AS (
SELECT YEAR(transaction_time) year, DATEPART(week, transaction_time) AS
week_number
	, COUNT( DISTINCT customer_id ) AS number_customer
FROM fact_table
JOIN dim_scenario AS scena ON fact_table.scenario_id = scena.scenario_id
WHERE category = 'Billing' AND status_id = 1 AND sub_category IN ('Electricity',
'Internet',  'Water')
GROUP BY YEAR(transaction_time), DATEPART(week, transaction_time)
-- ORDER BY year, week_number
)
SELECT *
	, SUM(number_customer) OVER ( PARTITION BY year ORDER BY
week_number ASC ) AS rolling_customer_year
FROM week_user
```

```
/* 2.2
Task: Based on the previous query, calculate the average number of customers for the
last 4 weeks in each observation week.
Then compare the difference between the current value and the average value of the
last 4 weeks.
*/

WITH fact_table AS (
SELECT * FROM fact_transaction_2019
UNION
SELECT * FROM fact_transaction_2020
)
, week_user AS (
SELECT YEAR(transaction_time) year, DATEPART(week, transaction_time) AS
week_number
        , COUNT( DISTINCT customer_id ) AS number_customer
FROM fact_table
JOIN dim_scenario AS scena ON fact_table.scenario_id = scena.scenario_id
WHERE category = 'Billing' AND status_id = 1 AND sub_category IN ('Electricity',
'Internet',  'Water')
GROUP BY YEAR(transaction_time), DATEPART(week, transaction_time)
)
-- Cần tính trung bình 4 tuần gần nhất --> trả kết quả về dòng hiện tại
SELECT *
        , AVG(number_customer) OVER ( PARTITION BY year ORDER BY
week_number ASC
                        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW )
AS avg_last_4_weeks
FROM week_user

-- Khi mà chúng ta cần tính rolling time window: WINDOW FUNCTION với ROWS
BETWEEN N/UNBOUDED PRECEDING/FOLLOWING AND CURENT ROW
-- PREDING: từ dòng hiện tại trở về trước
-- FOLLOWING: Từ dòng hiện tại trở về sau


-- Chúng ta chỉ có 1 pp tạo bảng trung gian: CTE:
---> bất tiện ở chỗ: câu lệnh càng dài càng cần nhiều CTE
```

---> Mình sẽ dùng bảng tạm: Local table
Cú pháp:

```
SELECT ...
INTO #local_table_name
FROM ...
JOIN ...
GROUP ...
```

```
WITH fact_table AS (
SELECT * FROM fact_transaction_2019
UNION
SELECT * FROM fact_transaction_2020
)
SELECT YEAR(transaction_time) year, DATEPART(week, transaction_time) AS
week_number
        , COUNT( DISTINCT customer_id ) AS number_customer
INTO #week_table
FROM fact_table
JOIN dim_scenario AS scena ON fact_table.scenario_id = scena.scenario_id
WHERE category = 'Billing' AND status_id = 1 AND sub_category IN ('Electricity',
'Internet')
GROUP BY YEAR(transaction_time), DATEPART(week, transaction_time)
```

```
SELECT *
        , AVG(number_customer) OVER ( PARTITION BY year ORDER BY
week_number ASC
                        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW )
AS avg_last_4_weeks
FROM #week_table
```

-- Bây giờ muốn thay đổi dữ liệu trong bảng local thì làm sao?

--> Phải xóa bảng --> INTO lại

```
DROP TABLE #week_table
```

-- phương pháp 2: Tạo bảng tạm : Tạo VIEWS -- sẽ hướng dẫn trong buổi 9

--> read:

-- LESSON 7: Correct homework and query notes --

```
-- 1.1
-- Basic retention curve
-- 1.1 A:

-- Way 1:
-- b1: Đi tìm tập customers 1/2019 mua Telco card thành công : 2,111 customers
WITH customer_list AS (
SELECT DISTINCT customer_id
FROM fact_transaction_2019 fact
JOIN dim_scenario sce ON fact.scenario_id = sce.scenario_id
WHERE sub_category = 'Telco Card' AND status_id = 1 AND
MONTH(transaction_time) = 1
)
, full_trans AS ( -- b2: Đi tìm tất cả giao dịch của tập trên : JOIN với fact_2019:
19,634 trans của tập trên
SELECT fact.*
FROM customer_list
JOIN fact_transaction_2019 fact
ON customer_list.customer_id = fact.customer_id
JOIN dim_scenario sce
ON fact.scenario_id = sce.scenario_id
WHERE sub_category = 'Telco Card' AND status_id = 1
)
-- b3: Đếm xem từng tháng có bao nhiêu khách hàng
SELECT MONTH(transaction_time) - 1 AS subsequence_month
        , COUNT( DISTINCT customer_id) AS retained_users
FROM full_trans
GROUP BY MONTH(transaction_time) - 1
ORDER BY subsequence_month

-- way2:
```

```
WITH period_table AS (
SELECT customer_id
         , transaction_id
         , transaction_time
         , MIN( MONTH(transaction_time)) OVER (PARTITION BY customer_id) AS
first_month
         , DATEDIFF(month, MIN( transaction_time) OVER (PARTITION BY
customer_id), transaction_time) AS subsequence_month
FROM fact_transaction_2019 fact
JOIN dim_scenario sce ON fact.scenario_id = sce.scenario_id
WHERE sub_category = 'Telco Card' AND status_id = 1
)
SELECT subsequence_month
         , COUNT( DISTINCT customer_id) AS retained_users
FROM period_table
WHERE first_month = 1
GROUP BY subsequence_month
ORDER BY subsequence_month


-- 1.1 B:
WITH period_table AS (
SELECT customer_id, transaction_id, transaction_time
         , MIN(transaction_time) OVER( PARTITION BY customer_id) AS first_time
         , DATEDIFF(month, MIN(transaction_time) OVER( PARTITION BY
customer_id), transaction_time) AS subsequent_month
FROM fact_transaction_2019 fact
JOIN dim_scenario sce ON fact.scenario_id = sce.scenario_id
WHERE sub_category = 'Telco Card' AND status_id = 1
)
, retained_user AS (
SELECT subsequent_month
         , COUNT( DISTINCT customer_id) AS retained_users
FROM period_table
WHERE MONTH(first_time) = 1
GROUP BY subsequent_month
-- ORDER BY subsequent_month
)
SELECT *
```

```
        , FIRST_VALUE(retained_users) OVER( ORDER BY subsequent_month) AS
original_users
        , MAX(retained_users) OVER() AS original_users_2
        , (SELECT COUNT(DISTINCT customer_id)
        FROM period_table
        WHERE MONTH(first_time) = 1) AS original_users_3
        , FORMAT(1.0*retained_users/FIRST_VALUE(retained_users) OVER(
ORDER BY subsequent_month ASC), 'p') AS pct_retained_users
FROM retained_user

-- 1.2 A
WITH period_table AS (
SELECT customer_id, transaction_id, transaction_time
        , MIN(MONTH( transaction_time)) OVER( PARTITION BY customer_id) AS
first_month
        , DATEDIFF(month, MIN(transaction_time) OVER( PARTITION BY
customer_id), transaction_time) AS subsequent_month
FROM fact_transaction_2019 fact
JOIN dim_scenario sce ON fact.scenario_id = sce.scenario_id
WHERE sub_category = 'Telco Card' AND status_id = 1
)
, retained_user AS (
SELECT first_month AS acquisition_month
        , subsequent_month
        , COUNT( DISTINCT customer_id) AS retained_users
FROM period_table
GROUP BY first_month , subsequent_month
-- ORDER BY acquisition_month, subsequent_month
)
SELECT *
        , FIRST_VALUE(retained_users) OVER( PARTITION BY acquisition_month
ORDER BY subsequent_month) AS original_users
        , FORMAT(1.0*retained_users/FIRST_VALUE(retained_users) OVER(
PARTITION BY acquisition_month ORDER BY subsequent_month), 'p') AS
pct_retained_users
INTO #retention_month -- lưu vào bảng local
FROM retained_user

SELECT * FROM #retention_month
```

```
-- DROP TABLE #retention_month
-- 1.2 B Pivot table

SELECT acquisition_month
        , original_users
        , "0", "1", "2", "3","4", "5", "6", "7","8", "9", "10", "11"
FROM (
        SELECT acquisition_month, subsequent_month, original_users,
pct_retained_users
        FROM #retention_month
) AS source_table
PIVOT ( -- MIN, MAX, AVG, SUM, COUNT
        MIN(pct_retained_users)
        FOR subsequent_month IN ("0", "1", "2", "3","4", "5", "6", "7","8", "9", "10",
"11")
) pivot_table
ORDER BY acquisition_month

-- User segmentation
-- RFM Segmenation
-- 2.1 Tính các chỉ số RFM

WITH fact_table AS ( -- 173,774 rows
        SELECT  fact_19.*
        FROM fact_transaction_2019 fact_19
        JOIN dim_scenario sce ON fact_19.scenario_id = sce.scenario_id
        WHERE sub_category = 'Telco Card' AND status_id = 1 -- 59,082 rows
UNION
        SELECT  fact_20.*
        FROM fact_transaction_2020 fact_20
        JOIN dim_scenario sce ON fact_20.scenario_id = sce.scenario_id
        WHERE sub_category = 'Telco Card' AND status_id = 1 -- 114,692 rows
)
, rfm_metric AS ( -- tính các metrics theo từng khách hàng
SELECT customer_id
        , DATEDIFF (day, MAX (transaction_time), '2020-12-31') AS recency --
khoảng cách từ
        , COUNT (DISTINCT CONVERT(varchar(10), transaction_time, 102)) AS
frequency -- đếm số ngày thanh toán, CONVERT về DATE
        , SUM(1.0*charged_amount) AS monetary
```

```
FROM fact_table
GROUP BY customer_id
)
, rfm_rank AS (
SELECT *
        , PERCENT_RANK() OVER ( ORDER BY recency ASC ) AS r_percent_rank
        , PERCENT_RANK() OVER ( ORDER BY frequency DESC ) AS
f_percent_rank
        , PERCENT_RANK() OVER ( ORDER BY monetary DESC ) AS
m_percent_rank
FROM rfm_metric
)
, rfm_tier AS (
SELECT *
        , CASE WHEN r_percent_rank > 0.75 THEN 4
        WHEN r_percent_rank > 0.5 THEN 3
        WHEN r_percent_rank > 0.25 THEN 2
        ELSE 1 END AS r_tier
        , CASE WHEN f_percent_rank > 0.75 THEN 4
        WHEN f_percent_rank > 0.5 THEN 3
        WHEN f_percent_rank > 0.25 THEN 2
        ELSE 1 END AS f_tier
        , CASE WHEN m_percent_rank > 0.75 THEN 4
        WHEN m_percent_rank > 0.5 THEN 3
        WHEN m_percent_rank > 0.25 THEN 2
        ELSE 1 END AS m_tier
FROM rfm_rank
)
, rfm_group AS (
SELECT *
        , CONCAT(r_tier, f_tier, m_tier) AS rfm_score -- tạo 1 cái score
FROM rfm_tier
) -- Step 3: Grouping these customers based on segmentation rules
, segment_table AS (
SELECT *
        , CASE
        WHEN rfm_score  =  111 THEN 'Best Customers'
        WHEN rfm_score LIKE '[3-4][3-4][1-4]' THEN 'Lost Bad Customer'
        WHEN rfm_score LIKE '[3-4]2[1-4]' THEN 'Lost Customers'
        WHEN rfm_score LIKE  '21[1-4]' THEN 'Almost Lost' -- sắp lost
        WHEN rfm_score LIKE  '11[2-4]' THEN 'Loyal Customers'
        WHEN rfm_score LIKE  '[1-2][1-3]1' THEN 'Big Spenders'
        WHEN rfm_score LIKE  '[1-2]4[1-4]' THEN 'New Customers'
        WHEN rfm_score LIKE  '[3-4]1[1-4]' THEN 'Hibernating'
        WHEN rfm_score LIKE  '[1-2][2-3][2-4]' THEN 'Potential Loyalists'
        ELSE 'unknown'
        END AS segment -- cố gắng ưu tiên tìm những segment muốn đầu tiên trước.
FROM rfm_group
)
SELECT
        segment
        , COUNT( customer_id) AS number_users
        , SUM( COUNT( customer_id)) OVER() AS total_users
        , FORMAT( 1.0*COUNT( customer_id) / SUM( COUNT( customer_id))
OVER(), 'p') AS pct
FROM segment_table
GROUP BY segment
ORDER BY number_users DESC
```