**Data warehousing**

What is a primary key?  
Is an int data type column or combination of columns that identifies all records. Normally these types of columns are also treated as **Identity columns.**

What is a foreign key?  
Is an int data type column or combination of columns that relates one table with another. It is used as a reference to the primary key of another table.  
The foreign and primary key have a parent child relationship where foreign key is the child and primary key is the parent.

How do you create an identity column?  
Is a column that is automatically computed by SQL server, starting as seed of 1 with increments of 1.  
Whenever an application creates a new row, SQL server generates an identity int value for this column.

What is the difference between a fact table and a dimension table?

Every business process provides two types of data:

* Numeric Data stored in Fact tables
* Descriptive Values stored in Dimension tables

Fact are measurements of business and Fact tables are used to calculate metrics such as:

* Quantity
* Sales Amount
* Total Earning
* Profit
* Margin
* Total Turnover

Dimensions are objects that describe facts.

What is a star schema? What are other schemas When and why would you use them?

**Star Schema:**

Is the simplest design technic for OLAP systems, consist on the division of data between fact tables and dimension tables.

Fact tables have numbers and dimension tables contain descriptive, values. The relationship between these two is done using foreign key values.

|  |  |
| --- | --- |
| **Fact\_country** | **Dim\_product** |
| Country\_ID | Product\_Id |
| Production | Product\_Name |
| Sales |  |
| Product\_Id |  |

**Snowflake:**

It looks the same as StarSchema, however the difference is in the Dimension tables:

|  |  |  |
| --- | --- | --- |
| **Fact\_country** | **Dim\_product** | **Dim\_Vendor** |
| Country\_ID | Product\_Id | VendorName |
| Production | Product\_Name | Product\_iD |
| Sales |  |  |
| Product\_Id |  |  |

|  |  |
| --- | --- |
| **Star Schema** | **SnowFlake** |
| Less Joins, simple queries. | Best to ensure data integrity and avoid duplication |
| Better speed and performance | Less redundancy and maintenance. |

What are the three types of slowly changing dimensions?

**SCD 0 the (information provided in the past is the source of truth and the rest will be discarded) –** If the source system changes the dimension value, our database will ignore this change.

Source System:

|  |  |  |
| --- | --- | --- |
| **From\_date** | **Previous\_ location** | **Current\_location** |
| 2020-01-01 | Barcelona | 2019-01-01 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **From\_date** | **Previous\_ location** | **Current\_location** |
| 0001 | 2020-01-01 | Barcelona | 2019-01-01 |
| 0001 | 2020-01-01 | Barcelona | 2019-01-01 |

Most data warehouses have SCD 1 or SCD2

**SCD 1 (to keep latest records only)** – stores current data keeping the latest record only and ignoring any previous value.

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **From\_date** | **Previous\_ location** | **Current\_location** |
| 0001 | 2020-01-01 | Barcelona | 2019-01-01 |

**SCD 2 (for historical) –** stores data in a transactional format using the dates and ids are to identify each transaction record.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **location** | **From\_date** | **To\_date** | **Current\_Flag** |
| 0001 | Madrid | 2019-01-01 | 2020-01-01 | No |
| 0001 | Barcelona | 2020-01-01 | 2021-01-01 | Yes |

**SCD 3 (for previous values) –** stores current and previous values as a column to be able to access these types of visualization.

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **From\_date** | **Previous\_ location** | **Current\_location** |
| 0001 | 2020-01-01 | Madrid | Barcelona |

**SCD 4 (for historical that changes every minute – used in banking what is a function SQLtransactions for bitcoins values) –** stores the information that comes from the source system in two tables, one contains the latest snapshot and the other contains historical.

|  |  |
| --- | --- |
| **Id** | **location** |
| 0001 | Madrid |
| 0002 | Barcelona |

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **location** | **From\_date** | **To\_date** |
| 0001 | Madrid | 2019-01-01 | 2020-01-01 |
| 0002 | Barcelona | 2020-01-01 | 2021-01-01 |

**General SQL**  
 **What is a function?**

Functions are database objects that contain sets of SQL statements. You can call these objects with input parameters to return single values or Tables. Functions can be used in stored procedures.

-- SQL functions can return a single scalar value or a table.

CREATE FUNCTION addTwoNumbers(@numberOne as int, @numberTwo as int)

RETURNS int

BEGIN

RETURN ( @numberOne + @numberTwo)  
END

---This function takes string 'a,b,c,d,e' and dumps the result into a table.

CREATE FUNCTION splitStringToTable(@string as nvarchar(100), @delimiter as nvarchar(1))

RETURNS @resultTable table( columnName nvarchar(10) not null)

AS

BEGIN

INSERT INTO @resultTable

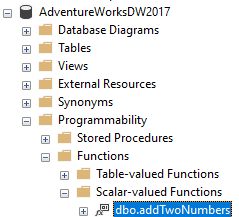
SELECT value

FROM string\_split(@string,@delimiter) where value <> ''

RETURN

END

Once created the function will be added to the database:

  
  
Then you can call the function by using:

SELECT dbo.addTwoNumbers(4,1)

SELECT \*

FROM dbo.splitStringToTable( 'a,b,c,d,e' , ',' )

**What is a stored procedure?**

Is a set of SQL statements saved in the database under a name. You can call the store procedure with the **EXECUTE *nameOfTheStoredProcedure*** command.

CREATE PROCEDURE spDatabaseTableNames

AS

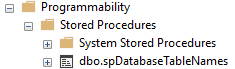
BEGIN

SELECT TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE='BASE TABLE';

END



EXECUTE spDatabaseTableNames

**What is a trigger?**

Triggers special store procedures that trigger when certain events happen on a particular table. These triggers are used to maintain the integrity of the data.  
  
There are two types of triggers: **INSERT, DELETE, UPDATE, INSTEAD OF**

**Trigger before the event:** Update or validate record values before they are saved to the database.  
  
**Example: Change the value of a column after the trigger.**

<https://www.youtube.com/watch?v=f6VWSlnHGCE>  
  
**Triggers after the event,** are used to access field values that are send by the system.

The records activated through the after trigger are read only. We cannot use after trigger if we want to update a record.  
**Example: Populate a new log table with information of the rows added after the trigger. The trigger should be inserted in table A and modification happens in the log table.**

<https://www.youtube.com/watch?v=k0S4P-a6d5w>

**What is a view?**

A view is a saved SQL query (virtual table).  
CREATE VIEW vWViewName

AS

SELECT \*

FROM hires\_terms;  
  
  
You can call that table and slice it as you want.

SELECT \*

FROM dbo.vWViewName;

**Why should we use views?**  
- The views reduce the complexity of the database schema, it is good for non IT users who can just call the view.   
- You can write a view for the user and give access to the user to the view for them to do their where statements.  
- Views can be used to implement security on row and columns, you can allow managers to have access to his department employees.  
- Can also be used to provide managers with **aggregated data**.

**What is a SQL Job?**  
Are a series of operations performed by a SQL Server Agent. Jobs can be run through command prompt applications, scripts or integration services packages.  
  
<https://docs.microsoft.com/en-us/sql/ssms/agent/create-jobs?view=sql-server-ver15>

In order to create a SQL job you need to:

* Connect to SQL server Agent (check that is running).
* Prepare the query to automate the job.
* Schedule the SQL job.

**What is are the different subsets of SQL?**

**Commands**  
  
DDL : Are used to define the database schema.  
DML: Deals with manipulation of Data.  
DCL : Deals with rights and permissions and other controls on the database.  
TCL : Includes commands which deal with the transaction of database.

**SQL syntax  
  
What is the difference between Delete and Truncate:**  
  
- TRUNCATE is a DDL command and DELETE is a DML command.  
  
The TRUNCATE Command:  
- Is extremely dangerous, removes all rows from the table.

- Is a very fast way to reset a table to an empty state.  
  
The DELETE Command:  
- Is used to remove records from a database.   
- It can be used in conjunction with WHERE command.  
- Is easier to roll back.

DELETE

FROM dbo.vWViewName  
WHERE empID = 1

**What is a group by?**

Is a statement used to aggregate functions in a summary:  
COUNT()

SUM()

AVG()

MIN() MAX()

**What is the difference between where and having?**

**WHERE** is a clause for filtering rows and applies on each and every row.

|  |  |  |
| --- | --- | --- |
| **Status** | **Item Type** | **Price** |
| Used | Bed | 500 |
| New | Sofa | 1500 |
| New | Table | 350 |
| Used | Chair | 20 |
| New | Bed | 1900 |
| Used | Sofa | 250 |

SELECT item\_type, sum(price) AS Total\_SUM

FROM tb  
WHERE price > 300  
GROUP BY item\_type

|  |  |
| --- | --- |
| **Status** | Total\_SUM |
| Used | 500 |
| New | 3750 |

**HAVING** is a clause for filtering based on aggregate functions.  
  
SELECT item\_type, sum(price) AS Total\_SUM  
FROM tb  
WHERE price > 300  
GROUP BY item\_type  
HAVING SUM(PRICE) > 600

|  |  |
| --- | --- |
| New | 3750 |

**Describe the different joins?**

**Cross Join:**  
The result is a cartesian product - multiplying the number of rows for each table.

|  |  |
| --- | --- |
| **Flight Departure** | **Flight Destination** |
| Vancouver | Madrid |
| New York | Chicago |
| Toronto | Barcelona |

|  |  |
| --- | --- |
| **Plane ID** | **Plane Name** |
| 1 | 747 Jumbo |
| 2 | Lear Jet |

SELECT \*

FROM table\_1 AS tb1  
CROSS JOIN table\_1 AS tb2  
  
This types of joins provide every possible combination between both tables.  
  
**Uses:**   
 - It is a quick way to generate a lot of data if there is the need to create sample data.  
 - In other cases when building data warehouses, some dimensions use cartesian products to provide every possible combination between attributes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Flight Departure** | **Flight Destination** | **Flight Departure** | **Flight Destination** |
| Vancouver | Madrid | 1 | 747 Jumbo |
| New York | Chicago | 1 | 747 Jumbo |
| Toronto | Barcelona | 1 | 747 Jumbo |
| Vancouver | **Flight Destination** | 2 | Lear Jet |
| New York | Madrid | 2 | Lear Jet |
| Toronto | Chicago | 2 | Lear Jet |

**Insert a column in a table:**

ALTER TABLE table\_name  
ADD column\_name datatype;

**Insert a row in a table:**

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

**SQL Operators:**

Arithmetic Operators: + - \* / %  
Comparison Operators : = != <>   
Logical Operators: BETWEEN, AND, ANY, EXIST, IN

**Difference between DROP and DELETE**  
The Drop Table statement is a Data Definition Language Command (DDL) that removes the named elements of the schema like relations, domains, constrains or entire schemas.  
  
DROP TABLE table\_name;

The DELETE statement is a Data Manipulation Language Command (DML) used to remove all tuples for a relation. When used with WHERE it removes all tuples that satisfy that statement.  
  
DELETE FROM employees

WHERE employeeID = 3;

**What is dbo. In this querry?**

SELECT \*

FROM dbo.vWViewName;  
  
dbo. Is the name of the schema or database owner. Is a user account implied permissions. Members of sysadmin are directly mapped to dbo.

**Change a value in a column to another**  
Change the value on the column event\_type from ‘H’ to ‘Hire’ in the table hires\_terms table.

|  |  |
| --- | --- |
| ID | Event Type |
| 1 | H |
| 2 | T |

UPDATE hires\_terms

SET event\_type = 'Hire'

WHERE event\_type = 'H'

|  |  |
| --- | --- |
| **ID** | Event Type |
| 1 | HIRE |
| 2 | T |

**Change the header name to another**

To rename a column we need to use the DDL (Database Definition Language) command ALTER.   
SQL server does not use ALTER to rename tables instead is used to   
  
**Add new Columns:**  
ALTER TABLE table\_name  
ADD column\_name datatype; **Delete Columns:**ALTER TABLE table\_name  
DROP COLUMN column\_name; **Modify datatypes of Columns:**ALTER TABLE table\_name  
ALTER COLUMN column\_name datatype;

**What is a Common Table Expressions (CTE)**  
Is a temporary result set, that can be referred with a SELECT, INSERT, UPDATE, DELETE.  
  
<https://www.youtube.com/watch?v=ZXB5b-7HJHk>  
  
Use them with Recursive Queries for hierarchies.  
<https://stackoverflow.com/questions/4740748/when-to-use-common-table-expression-cte>

It can be substitude for Views or Sored Procedures, you don’t need to store the definition in the metadata.  
  
**How do you sort in SQL?**  
SELECT \* FROM Customers  
ORDER BY Country, CustomerName;  
  
**How do you return unique values?**  
  
This gives you a list:  
  
SELECT DISTINCT Country   
FROM Customers;  
  
This gives you a count of distinct values on the list.  
  
SELECT COUNT(DISTINCT Country)   
FROM Customers;

**What is the difference between Union and Union All?**Union command is used to concatenate the rows of two SELECT statements. To be able to use these statements, **the schema** of the select statements must be the same.  
<https://www.youtube.com/watch?v=9w5uRCFOiTo>

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | Carlos |
| 2 | Sam |

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | Maria |
| 2 | Sam |

**UNION ALL:** returns all rows  
  
SELECT \* FROM tb1

UNION ALL

SELECT \* FROM tb2

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | Carlos |
| 2 | Sam |
| 1 | Maria |
| 2 | Sam |

**UNION**: returns all rows – duplicates and sorts the query (It is slower than Union ALL)

SELECT \* FROM tb1

UNION

SELECT \* FROM tb2

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | Carlos |
| 1 | Maria |

**WHAT IS THE DIFFERENCE BETWEEN TABLES AND FIELDS**

- Tables are collection of data organized in rows and columns, databases have n number of tables.  
  
- Fields are the columns of the tables, Tables are organized on n number of fields..

**WHAT IS THE DIFFERENCE BETWEEN CHAR and VARCHAR in SQL**

**CHAR** and **VARCHAR** are used for string datatrypes.  
- CHAR is used for strings with a specific length – CHAR(10)  
- VARCHAR is used for strings with a range of length. – VARCHAR(10) **WHAT ARE CONSTRAINTS**

Are rules used to specify the limits of the data.  
  
- NOT NULL :  
- UNIQUE :  
- DEFAULT : set up a default value when column contains ‘’ --- ‘unknown’   
- CHECK : ensures that values satisfy an specific condition.  
- INDEX : allows to generate data very quickly  
  
Example of an Index use case:

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Salary** |
| 1 | Carlos | 3500 |
| 2 | Sam | 2900 |
| 3 | Don | 1700 |
| 4 | Lynda | 5400 |

SELECT \* FROM tb

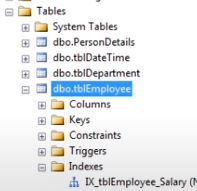
WHERE Salary BETWEEN 2300 AND 100000

This query is a ta table scans every single row and that is slow.

We should create an INDEX statement, the index will organize the table in ascending or descending order and find values from the specific point.  
  
CREATE INDEX IX\_tbname\_columnName  
ON tb (SALARY ASC)  
  
When running this command, you will be creating an index object stored in the specific table.

|  |  |
| --- | --- |
| **Salary** | **RowAddress** |
| 1700 | ID – xxx1 |
| 2900 | ID – xxx2 |
| 3500 | ID – xxx3 |
| 5400 | ID – xxx4 |

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Salary** |
| 1 | Carlos | 3500 |
| 2 | Sam | 2900 |
| 3 | Don | 1700 |
| 4 | Lynda | 5400 |

****

**Difference between clustered Index and non clustered index**

**CLUSTER INDEX:**  
  
- Alters the way that the records are stored in a database and sorts out rows by the column specified.  
- Is faster than the non cluster index.  
- A table may only have one CLUSTER INDEX.

**NON CLUSTER INDEX:**  
- Non-Cluster index does not alter the way the data is sorted but creates a separate object with a table which points back to the original table rows after searching.  
- A table may have multiple non cluster Indexes.  
  
A single table may have only one cluster index and many non clustered index.

**WHAT IS DATA INTEGRITY**

Data Integrity defines the **Accuracy** of the data, the **Consistency** and the **Integrity Constraints** to enforce business rules on the data.  
 **WHAT IS THE DIFFERENCE BETWEEN ENTITIES AND RELATIONSHJIPS  
  
Entities** real world information stored in the database. Tables are a type of entity.  
**Relationships** are links between entities.  
  
**WHAT IS DENORMALIZATION**

Technique to access data.  
Increases Performance and adds redundancy. Starchema is denormalized.

**DATES DML COMMANDS – To retrieve Dates:**

Write a SQL statement to display the current date:  
  
SELECT GETDATE()  
--Returns date and time:  
2020-06-04 18:02:23.743  
  
SELECT YEAR('01/31/2010')  
--Return 2010

SELECT DATENAME(Year, colA) , DATENAME(month , colA), DATENAME(day, colA)  
FROM FactResellerSales  
--Return 2010|December|29

**DATES DCL COMMANDS – To Set schema:**

DATE - YYYY-MM-DD   
DATETIME - YYYY-MM-DD HH:MI:S - 8 bytes storage  
SMALLDATETIME OR DATETIME2 - YYYY-MM-DD HH:MI:SS - 4 bytes TIMESTAMP - a unique epoch number

**CREATE a variable in TSQL**

Create a variable in TSQL

declare @test int;

set @test = 16;

print @test;

GO

**ROUND TSQL**

declare @test int;

set @test = 16;  
  
--This truncates all decimals from the value.

print ROUND( (@test \* 14.312341234123),2,0 );  
>> 229.000000000000  
  
--This rounds the value down to the 2nd decimal

print ROUND( (@test \* 14.312341234123),2,1 );  
>> 228.990000000000  
  
--This truncates all decimals from the value.  
print REPLACE(ROUND((@test \* 14.312341234123),2,1) , '0','');  
>> 228.99

**SSAS1**  
What is the difference between the tabular and multidimensional models?  
How do you set security based on roles?  
Give me an example of a situation where you created a many to many relationships.  
What is a measure? What is a dimension?

**SSIS**  
--It's hard to bullshit your way around SSIS. I suggest you tell them you don't have any experience, but you learned some on the side.  
What are different type of logs?  
What are different type of configuration variables?  
How do you automate an SSIS package?

**CREATE A NEW DATABASE FOR STORES:**

CREATE DATABASE db;  
USE bike\_stores;

**CREATE A TWO SQCHEMAS SALES AND PRODUCTION:**--THE SCHEEMA IS A DISTINCT NAME.SPACE SEPARATION TO FACILITATE MANAGEMENT, OWNERSHIP OF THE DATABASE OBJECTS..

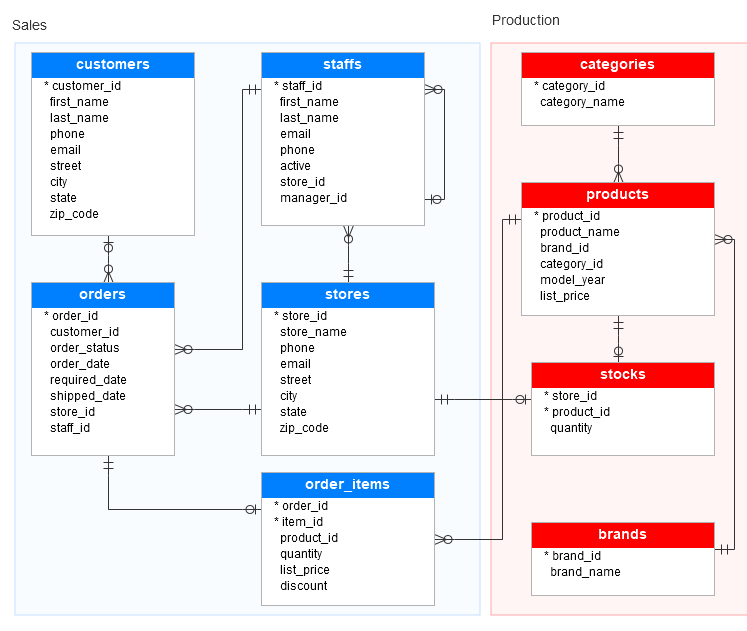
USE [bike\_stores]

GO

CREATE SCHEMA [sales]

GO

**CREATE A NEW DATABASE FOR STORES:**



**--CREATE OBJECTS FOR PRODUCTION SCHEMA:**  
  
CREATE TABLE production.categories (

category\_id INT IDENTITY (1, 1) PRIMARY KEY,

category\_name VARCHAR (255) NOT NULL

);

CREATE TABLE production.brands (

brand\_id INT IDENTITY (1, 1) PRIMARY KEY,

brand\_name VARCHAR (255) NOT NULL

);

CREATE TABLE production.products (

product\_id INT IDENTITY (1, 1) PRIMARY KEY,

product\_name VARCHAR (255) NOT NULL,

brand\_id INT NOT NULL,

category\_id INT NOT NULL,

model\_year SMALLINT NOT NULL,

list\_price DECIMAL (10, 2) NOT NULL,

FOREIGN KEY (category\_id) REFERENCES production.categories (category\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (brand\_id) REFERENCES production.brands (brand\_id) ON DELETE CASCADE ON UPDATE CASCADE

);  
  
  
**--CREATE OBJECT FOR SALES SCHEMA:**

CREATE TABLE sales.stores (

store\_id INT IDENTITY (1, 1) PRIMARY KEY,

store\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255),

street VARCHAR (255),

city VARCHAR (255),

state VARCHAR (10),

zip\_code VARCHAR (5)

);

**--CREATE OBJECT FOR PRODUCTION SCHEMA:**CREATE TABLE production.stocks (

store\_id INT,

product\_id INT,

quantity INT,

PRIMARY KEY (store\_id, product\_id),

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (product\_id) REFERENCES production.products (product\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**--CREATE OBJECTS FOR SALES SCHEMA:**

CREATE TABLE sales.customers (

customer\_id INT IDENTITY (1, 1) PRIMARY KEY,

first\_name VARCHAR (255) NOT NULL,

last\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255) NOT NULL,

street VARCHAR (255),

city VARCHAR (50),

state VARCHAR (25),

zip\_code VARCHAR (5)

);

CREATE TABLE sales.staffs (

staff\_id INT IDENTITY (1, 1) PRIMARY KEY,

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

email VARCHAR (255) NOT NULL UNIQUE,

phone VARCHAR (25),

active tinyint NOT NULL,

store\_id INT NOT NULL,

manager\_id INT,

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (manager\_id) REFERENCES sales.staffs (staff\_id) ON DELETE NO ACTION ON UPDATE NO ACTION

);

CREATE TABLE sales.orders (

order\_id INT IDENTITY (1, 1) PRIMARY KEY,

customer\_id INT,

order\_status tinyint NOT NULL,

-- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed

order\_date DATE NOT NULL,

required\_date DATE NOT NULL,

shipped\_date DATE,

store\_id INT NOT NULL,

staff\_id INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES sales.customers (customer\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (staff\_id) REFERENCES sales.staffs (staff\_id) ON DELETE NO ACTION ON UPDATE NO ACTION

);

CREATE TABLE sales.order\_items (

order\_id INT,

item\_id INT,

product\_id INT NOT NULL,

quantity INT NOT NULL,

list\_price DECIMAL (10, 2) NOT NULL,

discount DECIMAL (4, 2) NOT NULL DEFAULT 0,

PRIMARY KEY (order\_id, item\_id),

FOREIGN KEY (order\_id) REFERENCES sales.orders (order\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (product\_id) REFERENCES production.products (product\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

--DROP TABLES MUST BE IN ORDER IF THERE ARE FOREIGN KEYS CONSTRAINS OR MUST REMOVE THE CONSTRAIN.  
-- drop tables

DROP TABLE IF EXISTS sales.order\_items;

DROP TABLE IF EXISTS sales.orders;

DROP TABLE IF EXISTS production.stocks;

DROP TABLE IF EXISTS production.products;

DROP TABLE IF EXISTS production.categories;

DROP TABLE IF EXISTS production.brands;

DROP TABLE IF EXISTS sales.customers;

DROP TABLE IF EXISTS sales.staffs;

GO

DROP TABLE IF EXISTS sales.stores;

-- drop the schemas

DROP SCHEMA IF EXISTS sales;

DROP SCHEMA IF EXISTS production;