**Notes about SQL**

1. The SQL clause must be executable. MUST NOT contain **SYNTHETIC ERRORS**. If the assignment has a syntax error that causes the SQL clause not to be executed, it will receive a **score of 0**.
2. Select/Reset the **correct names of the columns** as shown in the answer (**LIKE the IMAGE showed in the result**).
3. **The output must be exactly the same as the result**. Also, since the post test engine is installed with **many other test cases**, the sql clause must be reasonable. DO NOT use unreasonable clauses solely for the purpose of getting the output shown in the resulting image.
4. **Necessary actions**:
   1. **Test SSMS** (start, connect to server, write and execute some basic commands) the day before test day
   2. **Check model (in system database) 🡪 empty** (avoid the case of creating a database, having tables automatically included)
   3. **When receive database file 🡪 View schema and diagram** to:
      1. Know the topic of the exercise
      2. Know the relationship of the tables --> Help to connect the tables
      3. Note the primary key and foreign key of the tables
   4. **Read** the **questions** and **see** their **results** carefully
5. **Some clues/words** to identify relational algebra operations to use in SQL clauses:
   1. Find all employees who work in project A ***or*** work in project B 🡪 UNION
   2. Find all employees who ***both*** work in project A ***and*** work in project B 🡪 INTERSECT
   3. Find all workers who ***work* in** project A **but do *not work*** in project B 🡪 EXCEPT
      * + Proceed as follows:

* Split the question into single ideas (separate at conjunctions such as: or, and, but/not)
* Write a query for each single idea (Note that the clauses must select columns to be consistent with each other). Because it must be the same column to perform set operations
* Concatenate the above simple queries by appropriate set operations

The following SQL statement returns the German cities (only distinct values) from both the "Customers" and the "Suppliers" table:

Example

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

* 1. If the data is of string type (VARCHAR, CHAR), when **comparing strings** you should **use ‘LIKE’**, limit the use of '='. Pay attention to **wildcards, string concatenation.**
     1. **Wildcard** characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column
     2. **String concatenation**
  2. **For Question 1**, when creating a table, **create table starting from the parent table first,** the table with the relationship number is 1.
  3. **For comparisons of date data types, the date should be extracted, and format strings should not be used for comparison:**
     1. Should: **YEAR(DateOfBirth) = ‘2000’**
     2. Should NOT: DateOfBirth LIKE ‘2000%’
  4. **For procedure/function parameters**, note that the **data type** declaration of the parameter/variable **must be exactly the same (including size)** as the data type of that object in the data table.
  5. For **GROUP BY**: **All columns before** the **function aggregate (MAX, MIN, COUNT, SUM, AVG)** **must be present in GROUP BY**

https://www.sqlservertutorial.net/sql-server-basics

|  |  |
| --- | --- |
| SELECT DISTINCT  column\_name1,  column\_name2 ,  ...  FROM  table\_name; | SELECT  city  FROM  sales.customers  ORDER BY  city;  SELECT DISTINCT  city  FROM  sales.customers  ORDER BY  city; |
| SELECT  select\_list  FROM  table\_name  WHERE  search\_condition; | SELECT  product\_id,  product\_name,  category\_id,  model\_year,  list\_price  FROM  production.products  WHERE  category\_id = 1 |
| SELECT  select\_list  FROM  table\_name  ORDER BY  column\_name | expression [ASC | DESC ]; | SELECT  first\_name,  last\_name  FROM  sales.customers  ORDER BY  first\_name; |
| SELECT TOP (expression) [PERCENT]  [WITH TIES]  FROM  table\_name  ORDER BY  column\_name; | SELECT TOP 10  product\_name,  list\_price  FROM  production.products  ORDER BY  list\_price DESC; |
| **operator** |  |
| AND | SELECT  \*  FROM  production.products  WHERE  category\_id = 1  AND list\_price > 400  AND brand\_id = 1  ORDER BY  list\_price DESC; |
| OR | SELECT  product\_name,  list\_price  FROM  production.products  WHERE  list\_price < 200  OR list\_price > 6000  ORDER BY  list\_price; |
| IN | SELECT  product\_name,  list\_price  FROM  production.products  WHERE  list\_price IN (89.99, 109.99, 159.99)  ORDER BY  list\_price; |
| BETWEEN | SELECT  product\_id,  product\_name,  list\_price  FROM  production.products  WHERE  list\_price BETWEEN 149.99 AND 199.99  ORDER BY  list\_price; |
| LIKE:   * % (percent) * \_ (underscore) * [ABC] (list of characters) * [^A-C] (similar to [ABC]) * NOT LIKE | SELECT  customer\_id,  first\_name,  last\_name  FROM  sales.customers  WHERE  last\_name LIKE 'z%'  ORDER BY  first\_name; |
| SELECT  select\_list  FROM  table\_name  GROUP BY  column\_name1,  column\_name2 ,...; | SELECT  customer\_id,  YEAR (order\_date) order\_year  FROM  sales.orders  WHERE  customer\_id IN (1, 2)  GROUP BY  customer\_id,  YEAR (order\_date)  ORDER BY  customer\_id; |
| SELECT  select\_list  FROM  table\_name  GROUP BY  group\_list  HAVING  conditions; | SELECT  customer\_id,  YEAR (order\_date),  COUNT (order\_id) order\_count  FROM  sales.orders  GROUP BY  customer\_id,  YEAR (order\_date)  HAVING  COUNT (order\_id) >= 2  ORDER BY  customer\_id; |

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| **%** | **Represents zero or more characters** | **bl% finds bl, black, blue, and blob** |
| **\_** | **Represents a single character** | **h\_t finds hot, hat, and hit** |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |
| - | Represents any single character within the specified range | c[a-b]t finds cat and cbt |

**Note on PE:**

**(PEA)**

**For the submission of your work:**

- Create a folder named **RollNo\_Name\_DBI202\_PaperNo**, e.g. se01245\_LongNT\_DBI202\_01. **Do not create** any **subfolder** in this folder. All solutions created will be located in the above folder.

- **Submit the folder** named **RollNo\_Name\_DBI202\_PaperNo**

- For each question, you are required to write a database script. Create a file with the name corresponding to the index of the question. For example, **for question 1**, we will create a file named **Q1.sql** and create a file **Q2.sql for question 2**. So, if you do 10 questions, **your folder must contain only 10 files Q1.sql, Q2.sql, Q3.sql, Q4.sql, Q5.sql, Q6.sql, Q7.sql, Q8.sql, Q9.sql and Q10.sql**.

- **Do not use** any commands having the **database name** such as create database, alter database, use [database name], *etc*.

- Your response must contain only necessary commands for answering the question. **Do not include** any other command. For example, if you are required to create a trigger/procedure, then your response should contain only commands for creating the corresponding trigger/procedure; all commands for **testing** the created trigger/procedure are forbidden.

- On completion, import your work by browsing to the above folder.

**- Note that:**

**+ You could use only SQL Server, SQL Server Management Studio, and paper or offline document in your computer.**

**+ If any of the previous requirements is not respected, your mark will be 0.**

**Question ERD:**

**Create one database and then write SQL statements to create tables from ERD**. In this database, all tables derived from the ERD given in the following picture with appropriate attributes, primary keys and foreign keys.

<

**Relationship:**

|  |  |  |
| --- | --- | --- |
| Cardinary | Action | Key |
| 1-1  M-M | Create a new table | The new table references the primary keys of two side tables, and set primary key |
| 1-M  M-1  *Note:*  *1: parent*  *M: child* | DO NOT create any new table | Child table references primary key of parent table |

Do not put a comma in the last line of the create table statement

Table with 2 attributes as primary key declares different from table with 1 attribute as primary key:

create table Dependants(

EmpCode varchar(20),

Number int,

Name nvarchar(50),

BirthDate Date,

Role nvarchar(30),

**primary key (EmpCode, Number)**

)

NOTE that when creating the SQL commands as request, **you MUST keep the name of tables, relationship, attributes and data type of attributes as SAME as given in the given ERD**.

Attributes with underline belong to the Primary Key of each entity.

Attributes which reference to the primary key of another table must have the same name as the attributes in the primary key of the referencing table.

When submitting the responses for this question, submit only SQL statements for creating tables with corresponding keys and foreign keys. **Do not use** **“create database”, “Alter database”, “use database\_name” or any statements using database’s name in your submission.**