Contents

General Knowledge

A **HAVING** clause is like a **WHERE** clause, but applies only to groups as a whole (that is, to the rows in the result set representing groups), whereas the WHERE clause applies to individual rows.

SQL Server

```
select
from dbo.additional_service

select
      srvc_name,
      min_participants
from dbo.additional_service

-- only top 3 rows
select top3
      srvc_name,
      min_participants
from dbo.additional_service

-- only top 3 rows
-- change column names
select top 3
      ColumnName1 = srvc_name,
      ColumnName1 = min_participants
from dbo.additional_service

-- only top 3 rows
-- change column names
select top 3
      srvc_name [Column Name 1],
      min_participants [Column Name 2]
from dbo.additional_service

--find the rows where the service name is 'Catering - Lunch'
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name = 'Catering - Lunch'

select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
--where srvc_name <> 'Catering - Lunch'
where srvc_name != 'Catering - Lunch'

--find the rows where the service name is 'Gift Basket Delivery - Small', 'Gift Basket
Delivery - Medium', or 'Gift Basket Delivery - Large'
select
```

```sql
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name IN (
      'Gift Basket Delivery - Small',
      'Gift Basket Delivery - Medium',
      'Gift Basket Delivery - Large')

--find the rows where the service name is not 'Gift Basket Delivery - Small', 'Gift
Basket Delivery - Medium', or 'Gift Basket Delivery - Large'
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name NOT IN (
      'Gift Basket Delivery - Small',
      'Gift Basket Delivery - Medium',
      'Gift Basket Delivery - Large')

--find the rows where the service name starts with 'Gift Basket Delivery'
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name LIKE 'Gift Basket Delivery%'

--find the rows where the service name contains 'Party'
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name LIKE '%Party%'

--find the rows where the service name does not start with 'Gift Basket Delivery'
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name NOT LIKE 'Gift Basket Delivery'

--find the rows where the price per person is between $75 and $125
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where per_person_price between 75 and 125

--find the rows where the price per person is less than $75
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where per_person_price < 75

--find the rows where the price per person is $125 or more
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where per_person_price >= 125


--when are NULL values returned?
select
```

```sql
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where min_participants IS NULL

--find the rows for services that require no more than 6 participants and costs no more
than $25 per person
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where min_participants <= 6 AND per_person_price <=25

--filter again to just show the catering services that meet the last criteria
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where min_participants <= 6
            AND per_person_price <=25
            AND srvc_name like '%catering%'

--find the rows for either catering services, gift baskets, or the Two Trees Tasting
Party
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where srvc_name like '%catering%'
            OR srvc_name like '%gift%'
            OR srvc_name = 'Two Trees Tasting Party'

--find the rows for services that cost less than $30 that are either catering or gift
baskets
select
      srvc_name,
      min_participants,
      per_person_price
from dbo.additional_service
where per_person_price < 30
            AND (srvc_name like '%catering%'
                  OR srvc_name like '%gift%'
                  OR srvc_name = 'Two Trees Tasting Party')

--join tables
select *
from dbo.product p INNER JOIN dbo.oil_flavor f
      on p.flavor_id = f.flavor_id

--join tables
select
      product_id,
      p.flavor_id,
      flavor_name,
      price
from dbo.product p INNER JOIN dbo.oil_flavor f
      on p.flavor_id = f.flavor_id

--join tables
select
      product_id,
      p.flavor_id,
      flavor_name,
      price,
      pt.* -- add all of the columns from product type table
from dbo.product p
INNER JOIN dbo.oil_flavor f
      on p.flavor_id = f.flavor_id
```

```
INNER JOIN dbo.product_type pt
      on p.product_type_id = pt.product_type_id

--join tables
select
      product_id,
      p.flavor_id,
      flavor_name,
      price,
      pt.* -- add all of the columns from product type table
from dbo.product p
INNER JOIN dbo.oil_flavor f
      on p.flavor_id = f.flavor_id
INNER JOIN dbo.product_type pt
      on p.product_type_id = pt.product_type_id
WHERE pt.product_type = 'Case'

--get the full name and order total for every customer named in Sydney

select full_name
from dbo.customer
where first_name = 'Sydney'

--get the full name and order total for every customer named in Sydney
select
      c.full_name,
      po.order_total
from dbo.customer c
INNER JOIN dbo.product_order po
      on c.customer_id = po.customer_id
where first_name = 'Sydney'

--get the full name and order total for every customer named in Sydney even if they
haven't purchased anything

select
      c.full_name,
      po.order_total
from dbo.customer c
LEFT OUTER JOIN dbo.product_order po
      on c.customer_id = po.customer_id
where first_name = 'Sydney'

--give the names of all of the customers who live in Boston
select *
from dbo.city_state_zip
where city_name = 'Boston'

select full_name
from dbo.customer c
where city_state_zip_id IN (9,10)

--give the names of all of the customers who live in Boston

select full_name
from dbo.customer c
where city_state_zip_id IN
      (
      select city_state_zip_id
      from dbo.city_state_zip
      where city_name = 'Boston'
      )
--return the full_name and spending tier for orders
--spending tier '1 - low' = order less than $20
--spending tier '2 - medium' = order total between $20 and $100
--spending tier '3 - high' = order total higher than $100
--else = '0 - no recent orders'
select
      c.full_name,
      order_total,
      spending_tier = case
            when po.order_total <20
```

```
                then 'low'
            when po.order_total between 20 and 100
                then 'medium'
            when po.order_total >100
                then 'high'
            else 'no orders'
      end
from dbo.customer c
 JOIN dbo.product_order po
      on c.customer_id = po.customer_id

-- aggregates
select
      c.full_name,
      yr = year(o.order_date),
      lowest_order_total = MIN(o.order_total),
      highest_order_total = MAX(o.order_total),
      average_order_total = AVG(o.order_total),
      count_order_total = COUNT(o.order_total)
from dbo.customer c
inner join dbo.product_order o
      on c.customer_id = o.customer_id
group by c.full_name, year(o.order_date)
order by c.full_name desc
```

T-SQL

```
DROP VIEW MyView;

CREATE VIEW MyView AS
SELECT SupplierCategoryID,    SupplierCategoryName
FROM Purchasing.SupplierCategories;
GO

SELECT * FROM MyView;

-- Rename a column in the CustomerTransactions table
EXEC sp_rename 'Sales.CustomerTransactions.AmountExcludingTax', 'PreTaxTotal', 'COLUMN';
GO

-- Modify the view to fix the column reference
ALTER VIEW Sales.OutstandingBalance
AS
SELECT
    CustomerTransactionID,
    CustomerID,
    TransactionDate,
    PreTaxTotal AS AmountExcludingTax,  -- Alias added to display old name in results
    TaxAmount,
    OutstandingBalance
FROM Sales.CustomerTransactions
WHERE OutstandingBalance > 0;
GO

ALTER VIEW Sales.OutstandingBalance
WITH SCHEMABINDING -- lock the view, so the changes to the table will no longer be
allowed. In order to make a change the view needs to be altered to dropped first
AS
SELECT
    CustomerTransactionID AS 'Transaction Number',
    CustomerID AS 'Customer Number',
    TransactionDate AS 'Order Date',
    PreTaxTotal AS 'Amount Before Tax',
    TaxAmount AS 'Tax Due',
    OutstandingBalance AS 'Balance Due'
FROM Sales.CustomerTransactions
WHERE OutstandingBalance > 0
;
GO

-- Some functions require multiple parameters or arguments
SELECT DISTINCT TOP 10
    InvoiceDate,
    FORMAT(InvoiceDate, 'd') AS FormattedDate -- standard date formatting for local
culture
FROM Sales.Invoices;

-- Functions can be nested together
SELECT FORMAT(GETDATE(), 'MMMM dd, yyyy "at" hh:mm');


-- Creating scalar valued functions
USE WideWorldImporters;
GO
-- Basic function to square a number
CREATE FUNCTION Application.SquareNumber (@InputNumber AS INT)
RETURNS INT
AS
BEGIN
    DECLARE @Output INT;
    SET @Output = @InputNumber * @InputNumber;
    RETURN @Output;
END;
GO
-- Use the function
```

```sql
PRINT Application.SquareNumber(5);
-- Use the function with data from the database
SELECT TOP 10 OrderID AS 'A Number',
    Application.SquareNumber(OrderID) AS 'The Number Squared'
FROM Sales.Orders;


-- IF ELSE
CREATE OR ALTER FUNCTION Application.EvenOdd (@InputNumber INT)
RETURNS char(10)
AS
BEGIN
    DECLARE @Output char(10);
    BEGIN IF @InputNumber % 2 = 0
        SET @Output = 'Even'
        ELSE SET @Output = 'Odd'
    END;
    RETURN @Output;
END;
GO
-- Use the IF ELSE function
SELECT Application.EvenOdd(2) AS '2',
    Application.EvenOdd(3) AS '3';
GO

SELECT OrderID AS InputNumber,
    Application.EvenOdd(OrderID) AS 'Even or Odd'
FROM Sales.Orders;
GO


-- CASE statement
CREATE OR ALTER FUNCTION Application.Weekend (@Day char(10))
RETURNS char(3)
AS
BEGIN
    DECLARE @Output char(3);
    SET @Output =
        CASE WHEN @Day = 'Saturday' THEN 'Yes'
            WHEN @Day = 'Sunday' THEN 'Yes'
            ELSE 'No'
        END;
    RETURN @Output;
END;
GO
-- Use the CASE function
SELECT Application.Weekend('Sunday') AS Sun,
    Application.Weekend('Monday') AS Mon,
    Application.Weekend('Tuesday') AS Tue,
    Application.Weekend('Wednesday') AS Wed,
    Application.Weekend('Thursday') AS Thur,
    Application.Weekend('Friday') AS Fri,
    Application.Weekend('Saturday') AS Sat
;


-- Write a query that retrieves the required information
-- This query will serve as the model for the function
        -- Turn the query into a table-valued function
CREATE OR ALTER FUNCTION Sales.LastOrder (@CustomerID AS INT)
RETURNS TABLE
AS RETURN
SELECT
    Orders.OrderID AS [Order Number],
    Orders.CustomerID AS [Customer Number],
    Customers.CustomerName AS [Customer Name],
    Orders.OrderDate AS [Order Date],
    Orders.ExpectedDeliveryDate AS [Delivery Date],
    OrderLines.OrderLineID AS [Line Number],
    OrderLines.Description AS [Product Description]
FROM Sales.Orders
    INNER JOIN Sales.OrderLines ON Orders.OrderID = OrderLines.OrderID
```

```sql
    INNER JOIN Sales.Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Orders.OrderID =
    (SELECT TOP 1 Orders.OrderID
     FROM Sales.Orders
     WHERE Orders.CustomerID = @CustomerID
     ORDER BY Orders.OrderID DESC)
;
GO
-- Test out the table-valued function
SELECT * FROM Sales.Lastorder(123);
SELECT * FROM Sales.LastOrder(828);


-- Basic stored procedure
CREATE OR ALTER PROCEDURE Application.uspViewEmployees
AS
SELECT PersonID, FullName, IsEmployee, IsSalesperson
FROM Application.People
WHERE IsSalesperson = 1;
GO
-- Execute a stored procedure
EXECUTE Application.uspViewEmployees;
GO


-- Update the stored procedure to mask column names
CREATE OR ALTER PROCEDURE Application.uspViewEmployees
AS
SELECT PersonID AS 'ID Number',
    FullName AS 'Name',
    'Employee' AS Status,
    CASE WHEN IsSalesperson = 1 THEN 'Salesperson'
         WHEN IsSalesperson = 0 THEN 'Not Salesperson'
    END AS Position
FROM Application.People
GO


-- Create a procedure to insert new rows
CREATE OR ALTER PROCEDURE Warehouse.uspInsertColor (@Color AS nvarchar(100))
AS
    DECLARE @ColorID INT
    SET @ColorID = (SELECT MAX(ColorID) FROM Warehouse.Colors)+1;
    INSERT INTO Warehouse.Colors (ColorID, ColorName, LastEditedBy)
        VALUES (@ColorID, @Color, 1);
    SELECT * FROM Warehouse.Colors
        WHERE ColorID = @ColorID
        ORDER BY ColorID DESC;
;
GO
-- Test the stored procedure
EXEC Warehouse.uspInsertColor @Color = 'Periwinkle Blue'; -- or remove variable name if
supplied in order

SELECT * FROM Warehouse.Colors
ORDER BY ColorID DESC;
GO


-- Challenge Two Complete
USE WideWorldImporters;
GO
-- Create an audit table for customer accounts
CREATE TABLE Sales.CustomerAccountAudit (
    AuditID INT IDENTITY PRIMARY KEY,
    CustomerID INT,
    ReviewDate datetime2
);
GO
-- View existing data
SELECT * FROM Sales.Customers;
SELECT * FROM Sales.Orders;
```

```sql
GO
-- Write a stored procedure to:
-- 1) view information from Sales.Customers
-- 2) view information from Sales.Orders
-- 3) write a row to Sales.CustomerAccountAudit to log activity
CREATE OR ALTER PROCEDURE Sales.SalesInfo (@Customer AS INT)
AS
SELECT CustomerID, CustomerName, PhoneNumber
      FROM Sales.Customers
      WHERE CustomerID = @Customer;
SELECT OrderID, CustomerID, OrderDate
      FROM Sales.Orders
      WHERE CustomerID = @Customer;
INSERT INTO Sales.CustomerAccountAudit (CustomerID, ReviewDate)
      VALUES (@Customer, GETDATE());
;
GO
-- Test the stored procedure
EXEC Sales.SalesInfo 915;
```

MySQL

```
SHOW DATABASES;

create database tutorial;

-- create a table
USE tutorial;
CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT NOT NULL,
    FirstName VARCHAR(50) NOT NULL DEFAULT 'Jane',
    LastName VARCHAR(50) NOT NULL DEFAULT 'Doe',
    Email VARCHARACTER(50) NOT NULL UNIQUE,
    Phone VARCHAR(20),
    Address VARCHAR(50),
    City VARCHAR(50),
    State CHAR(2),
    Zipcode CHAR(5),
    CHECK (State = 'NY'),
    PRIMARY KEY (CustomerID)
);

USE HPlus;
SELECT
    OrderID, TotalDue, CustomerID
FROM
    Orders
WHERE
    TotalDue > 300;

USE tutorial;
CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT NOT NULL,
    FirstName VARCHAR(50) NOT NULL DEFAULT 'Jane',
    LastName VARCHAR(50) NOT NULL DEFAULT 'Doe',
    Email VARCHARACTER(50) NOT NULL UNIQUE,
    Phone VARCHAR(20),
    Address VARCHAR(50),
    City VARCHAR(50),
    State CHAR(2),
    Zipcode CHAR(5),
    CHECK (State = 'NY'),
    PRIMARY KEY (CustomerID)
);

USE HPlus;
SELECT Orders.OrderID, Customer.FirstName, Orders.status
FROM Orders
INNER JOIN Customer ON Orders.CustomerID = Customer.Customerid;

USE HPlus;

SELECT Salesperson.FirstName, Salesperson.LastName, AVG(Orders.TotalDue) AS 'Average
Sales'
FROM Salesperson
INNER JOIN Orders ON Salesperson.SalespersonID = Orders.SalespersonID
GROUP BY Salesperson.SalespersonID;

USE HPlus;
SELECT Salesperson.FirstName, Salesperson.LastName, AVG(Orders.TotalDue) AS AverageSales
FROM Salesperson
INNER JOIN Orders ON Salesperson.SalespersonID = Orders.SalespersonID
GROUP BY Salesperson.SalespersonID
ORDER BY AverageSales DESC
LIMIT 5; -- top 5

SELECT a.title AS album, a.artist, t.track_number AS seq, t.title, t.duration AS secs
  FROM album AS a
  JOIN (
    SELECT DISTINCT album_id, track_number, duration, title
```

```
    FROM track
    WHERE duration <= 90
  ) AS t
    on t.album_id = a.id
  ORDER BY a.title, t.track_number
;

-- 03 Creating a view

USE album;
SELECT id, album_id, title, track_number, duration DIV 60 AS m, duration MOD 60 AS s FROM
track;

CREATE VIEW trackView AS
  SELECT id, album_id, title, track_number, duration DIV 60 AS m, duration MOD 60 AS s
FROM track;
SELECT * FROM trackView;

SELECT a.title AS album, a.artist, t.track_number AS seq, t.title, t.m, t.s
  FROM album AS a
  JOIN trackView AS t
    ON t.album_id = a.id
  ORDER BY a.title, t.track_number
;

SELECT a.title AS album, a.artist, t.track_number AS seq, t.title,
    CONCAT(t.m, ':', SUBSTR(CONCAT('00', t.s), -2, 2)) AS duration
  FROM album AS a
  JOIN trackView AS t
    ON t.album_id = a.id
  ORDER BY a.title, t.track_number
;

DROP VIEW IF EXISTS trackView;

-- 04 Joined view

USE album;
SELECT a.artist AS artist,
    a.title AS album,
    t.title AS track,
    t.track_number AS trackno,
    t.duration DIV 60 AS m,
    t.duration MOD 60 AS s
  FROM track AS t
    JOIN album AS a
      ON a.id = t.album_id
  ORDER BY a.artist, t.track_number
;

CREATE VIEW joinedAlbum AS
    SELECT a.artist AS artist,
        a.title AS album,
        t.title AS track,
        t.track_number AS trackno,
        t.duration DIV 60 AS m,
        t.duration MOD 60 AS s
      FROM track AS t
          JOIN album AS a
            ON a.id = t.album_id
    ORDER BY a.artist, t.track_number
    ;

SELECT * FROM joinedAlbum;
SELECT * FROM joinedAlbum WHERE artist = 'Jimi Hendrix';

SELECT artist, album, track, trackno,
    CONCAT(m, ':', SUBSTR(CONCAT('00', s), -2, 2)) AS duration
    FROM joinedAlbum;

DROP VIEW IF EXISTS joinedAlbum;
```

WEB3

https://www.w3schools.com/python/default.asp

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;

UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

SELECT * FROM Customers
LIMIT 3; -- MySQL

SELECT TOP 3 * FROM Customers; -- SQL Server/MS Access

SELECT * FROM Customers
WHERE CustomerName LIKE '_r%'; -- r in second position

SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%'; -- begins with a and has length of at least 3

SELECT * FROM Customers
WHERE ContactName LIKE 'a%o'; -- begins with a and ends with o

SELECT * FROM Customers
WHERE CustomerName NOT LIKE 'a%'; -- not starts with a

SELECT * FROM Customers
WHERE City LIKE '_ondon'; -- selects all customers with a City starting with any
character, followed by "ondon":

SELECT * FROM Customers
WHERE City LIKE 'L_n_on';

SELECT * FROM Customers
WHERE City LIKE '[bsp]%'; -- select all customers with a City starting with "b", "s", or
"p”

SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';

SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');

SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);

SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;

SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;

SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

```sql
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;

SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address
FROM Customers; -- SQL Server
creates an alias named "Address" that combine four columns (Address, PostalCode, City and
Country)

SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address
FROM Customers; -- MySQL

SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;

SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;

SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;

SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;

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

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

SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;

SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName;
```

```sql
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;

SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);

SELECT ProductName
FROM Products
WHERE ProductID = ANY
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10); -- The following SQL statement lists the ProductName if it finds
ANY records in the OrderDetails table has Quantity equal to 10 (this will return TRUE
because the Quantity column has some values of 10):
SELECT ProductName
FROM Products
WHERE ProductID = ALL
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10); --The following SQL statement lists the ProductName if ALL the
records in the OrderDetails table has Quantity equal to 10. This will of course return
FALSE because the Quantity column has many different values (not only the value of 10):

SELECT * INTO CustomersBackup2017
FROM Customers;
 -- The following SQL statement creates a backup copy of Customers:

SELECT * INTO CustomersBackup2017 IN 'Backup.mdb'
FROM Customers; -- The following SQL statement uses the IN clause to copy the table into
a new table in another database:

SELECT CustomerName, ContactName INTO CustomersBackup2017
FROM Customers;

SELECT * INTO CustomersGermany
FROM Customers
WHERE Country = 'Germany';

INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
WHERE Country='Germany'; -- The following SQL statement copies "Suppliers" into
"Customers" (the columns that are not filled with data, will contain NULL):

SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;

SELECT ProductName, UnitPrice * (UnitsInStock + ISNULL(UnitsOnOrder, 0))
FROM Products; -- The SQL Server ISNULL() function lets you return an alternative value
when an expression is NULL:

SELECT ProductName, UnitPrice * (UnitsInStock + COALESCE(UnitsOnOrder, 0))
FROM Products; -- or we can use the COALESCE() function, like this:

SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, 0))
FROM Products; -- The MySQL IFNULL() function lets you return an alternative value if an
expression is NULL:
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;

EXEC SelectAllCustomers;
```

```sql
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND PostalCode = @PostalCode
GO;

EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';

CREATE DATABASE testDB;

DROP DATABASE testDB;

BACKUP DATABASE testDB
TO DISK = 'D:\backups\testDB.bak';

CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);

CREATE TABLE TestTable AS
SELECT customername, contactname
FROM customers;

DROP TABLE Shippers;

TRUNCATE TABLE table_name; -- to delete data in the table

ALTER TABLE Customers
ADD Email varchar(255);

ALTER TABLE Customers
DROP COLUMN Email;

ALTER TABLE table_name
ALTER COLUMN column_name datatype; -- SQL Server

ALTER TABLE table_name
MODIFY COLUMN column_name datatype; -- MySQL

CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
); -- SQL Server

CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
); -- SQL Server

CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
); -- SQL Server

CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
); --SQL Server
```

```sql
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
); -- MySQL

CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
); --My SQL

CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
); -- MySQL

CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
); -- MySQL

CREATE OR REPLACE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName, City
FROM Customers
WHERE Country = 'Brazil';

DROP VIEW [Brazil Customers];
```

Budowa i eksploatacja baz danych

- select department_id, employee_id, last_name
  - from employees
- odczytaj nazwiska (last_name) i imiona (first_name) wszystkich pracowników w firmie
  - select*
  - from departments
- odczytaj nr departamentów (department_id) oraz nazwy departamentów wszystkich departamentów (departments)
  - select last_name, first_name
  - from employees
- odczytaj dolne i górne granice poszcz stawek (lowest_sal, highest_sal)
  - select lowest_sal, highest_sal
  - from JOB_GRADES
- odczytaj nazwiska (last_name), daty zatrudnienia (hire_date), pensje (salary) wszystkich pracowników (employees)
  - select last_name, hire_date, salary
  - from employees
- Oczytaj nazwiska, imiona oraz miesięczne pensje dla wszystkich pracowników. Wyniki posortuj wg pensji malejąco, a w ramach tej samej pensji - alfabetycznie po nazwisku.
  - select last_name, first_name, salary
  - from employees
  - order by salary desc, last_name;
- Utwórz raport zawierający nazwiska, imiona oraz roczne pensje pracowników z departamentu 10
  - select last_name, first_name, salary*12
  - from employees
  - where department_id = 10;
- Znajdź wszystkich pracowników (nazwiska, imiona, miesięczne pensje oraz nry departamentów), których roczna pensja mieści się w przedziale pomiędzy 50000 a 100000
  - select last_name, first_name, salary, department_id
  - from employees
  - where salary*12 between 50000 and 100000;
- wyświetl wszystkie dane wszystkich departamentów
  - select *
  - from departments;
- wydrukuj nazwiska wszystkich pracowników, których nazwiska zaczynają się na literę 'A'
  - select last_name
  - from employees
  - where last_name like 'A%'
- Zbuduj raport, który zawiera nr departamentu, nazwisko, imię, miesięczną pensję oraz wartość rocznej prowizji dla wszystkich pracowników. Wynik posortuj wg nru departamentu oraz rocznej pensji
  - select department_id, last_name, first_name, salary, salary*12*commission_pct
  - from employees;
- Zbuduj raport zawierający nazwiska, imiona oraz roczne pensje dla wszystkich pracowników z departamentu 20
  - select last_name, first_name, salary*12
  - from employees
  - where department_id = 20;
- Znajdź wszystkich pracowników, których nazwiska kończą się znakiem 'T'
  - select last_name
  - from employees
  - where last_name like '%t';
- SELECT DISTINCT department_id, salary
  - FROM employees;
- select department_id, last_name, first_name, salary
  - from employees
  - order by department_id, salary
- select department_id, last_name, salary
  - from employees
  - order by salary desc, last_name
- select employee_id, last_name, first_name
  - from employees
  - where last_name like 'H%';
- SELECT last_name, salary
  - FROM employees
  - WHERE salary > 10000;

- select employee_id, last_name, first_name, salary
  - from employees
  - where salary between 1000 and 4000;
- select employee_id, last_name, first_name, salary
  - from employees
  - where salary not between 1000 and 4000;
- select last_name, salary, department_id
  - from employees
  - where department_id in (10,20,30);
- select last_name, salary, department_id
  - from employees
  - where department_id not in (10,20,30);
- select employee_id, last_name, first_name, comission_pct, salary
  - from employees
  - where comission_pct IS NULL;
- select employee_id, last_name, first_name, comission_pct, salary
  - from employees
  - where comission_pct IS NOT NULL;
- Aliasy kolumn pozwalają nam zbudować zastępcze nagłówki kolumn w raportach.
  - select department_id id_of_the_department, department_name name_of_the_department
  - from departments;
- Funkcja ta umożliwia obsługę wartości NULL. Jej potrzeba wynika z faktu, że jeśli w skład dowolnego wyrażenia arytmetycznego wchodzi wartość NULL, to całe to wyrażenie również przyjmuje wartość NULL.
  - select last_name, salary*12 + nvl(salary*commission_pct,0)
  - from employees;
- SELECT employee_id, last_name, first_name, salary
  - FROM employees
  - WHERE salary > 5000 AND last_name LIKE 'S%';
- SELECT employee_id, last_name, first_name, salary
  - FROM employees
  - WHERE salary > 5000 OR last_name LIKE 'D%';
- SELECT employee_id, last_name, first_name, salary, commission_pct
  - FROM employees
  - WHERE commission_pct IS null
- SELECT employee_id, last_name, first_name, salary, commission_pct
  - FROM employees
  - WHERE commission_pct IS NOT null
- Odczytaj nazwiska, imiona i pensje wszystkich pracowników z departamentu 10 z pensją większą, niż 2000
  - SELECT last_name, first_name, salary, department_id
  - FROM employees
  - where department_id in (20) AND salary>2000;
- Odczytaj id pracowników, nazwiska imiona i pensje wszystkich pracowników zatrudnionych poza departamentem 10 oraz z pensją mniejszą lub równą niż 5000
  - SELECT EMPLOYEE_ID, last_name, first_name, salary, department_id
  - FROM employees
  - WHERE department_id NOT in (20) AND salary >5000
- Który pracownik (nazwisko i imię) nie ma managera ?
  - SELECT last_name, first_name
  - FROM employees
  - WHERE MANAGER_ID IS null
- Odczytaj id, nazwiska, imiona, daty zatrudnienia, pensje i procenty prowizji wszystkich pracowników spoza departamentów 10 i 20, którzy posiadają jakąkolwiek prowizję
  - SELECT EMPLOYEE_ID, last_name, first_name, HIRE_DATE, salary, COMMISSION_PCT
  - FROM employees
  - WHERE department_id NOT in (20) AND department_id NOT in (10) AND COMMISSION_PCT is not null
- Odczytaj dane pracowników (nazwiska, imiona i stanowiska pracy) wszystkich pracowników, którzy NIE SĄ zatrudnieni na stanowisku CLERK
  - SELECT JOB_ID, last_name, first_name
  - FROM employees
  - WHERE JOB_ID != 'ST_CLERK'
- Odczytaj numery oraz nazwy departamentów, których numery NIE mieszczą się na liście 10, 20, 30
  - SELECT JOB_ID, last_name, first_name
  - FROM employees
  - WHERE JOB_ID != 'ST_CLERK'
- Odczytaj stawki płacowe (tabela job_grades) dla których minimalna pensja jest większa lub równa 4000

        - select grade_level
        - from job_grades
        - where lowest_sal >=4000
    - odczytaj Nazwiska, imiona orac roczne dochody pracownikow dla ktorych procent prowizji jest niepusty lub sa zatrudnieni w departamentach 10 lub 20 (miesiecczna pensja + miesiecszna pensja*procent_prowizji)* 12 + bonus
        - SELECT last_name, first_name, 12*salary + nvl(commission_pct,0)*salary + nvl(bonus,0)
        - from employees
        - where commission_pct IS NOT null AND (department_id in (10) OR department_id in (20))
    - odczytaj nazwiska, imiona, stanowiska pracy, nry departamentów oraz pensje pracowników, którzy
        - mają nazwiska rozpoczynające się na 'S' lub zarabiają poniżej 5000
            - SELECT last_name, first_name, job_id, department_id, salary
            - from employees
            - where last_name like 'K%' AND salary<5000
        - pracują na stanowisku ST_CLERK i są zatrudnieni w departamencie 20 lub 30
            - SELECT last_name, first_name, job_id, department_id, salary
            - from employees
            - where job_id = 'ST_CLERK' AND (department_id =10 or department_id =20)
    - Który pracownik ma nazwisko składające się z czterech liter i zaczynające się na K
        - SELECT last_name
        - from employees
        - where LENGTH(last_name) =4 AND last_name like 'K%';
    - Odczytaj nazwiska i stanowiska pracy pracowników pracujących w departamencie 50, z pensją powyżej 3000
        - SELECT last_name, job_id
        - FROM employees
        - where department_id in (50) AND salary>3000;
    - Odczytaj nazwiska, imiona, stanowiska pracy i pensje pracowników zatrudnionych na stanowisu SA_REP
        - SELECT last_name, first_name, salary
        - FROM employees
        - WHERE JOB_ID = 'SA_REP'
    - konwertuje swój parametr param do dużych liter. Znaki nie będące literami nie podlegają konwersji
        - SELECT last_name, upper(last_name)
        - FROM employees;
    - konwertuje swój parametr do małych liter. Znaki nie będące literami nie podlegają konwersji
        - SELECT last_name, upper(last_name), lower(last_name)
        - FROM employees
        - WHERE lower(last_name) = 'king';
    - każda pierwsza litera każdego słowa w ciągu znakowym param jest konwertowana do dużej, pozostałe litery - do małych, Znaki nie będące literami nie podlegają konwersji
        - SELECT initcap(last_name||' is an employee of department '||department_id)
        - FROM employees;
    - varchar2 substr(varchar2 string, number start_position [, number length]) wycina podciąg znakowy z ciągu string
        - select substr(last_name,2)
        - from employees
        - where upper(substr(last_name,2,1)) = 'I';
    - zwraca liczbę znaków (długość) ciągu znakowego param
        - select length(last_name), last_name
        - from employees
        - where length(last_name) > 4;
    - select last_name, replace(last_name, 'K','L')
        - from employees;
    - ciąg znakowy string1 zostanie wypełniony z lewej strony do długości padded_length znakami pad_string. Jesli parametr pad_string zostanie pominięty, to lpad będzie wypełniał spacjami
        - select last_name, lpad(last_name,20,'*')
        - from employees;
    - select last_name, lpad(last_name,20)
        - from employees;
    - ciąg znakowy string1 zostanie wypełniony z prawej strony do długości padded_length znakami pad_string. Jesli parametr pad_string zostanie pominięty, to lpad będzie wypełniał spacjami
        - select last_name, rpad(last_name,20,'*')
        - from employees;

- number round (number value [, number decimal_places])
  - select last_name, salary, round(salary*1.50,-3)
  - from employees;
- number trunc (number value [, number decimal_places])
  - select last_name, salary, trunc(salary*1.50,-3), round(salary*1.50,-3)
  - from employees;
- wydrukuj nazwiska, imiona wszystkich pracowników z departamentu 10 oraz ich pensje zwiększone o 10% i zaokrąglone do 2 miejsc po przecinku
  - select last_name, salary, round(salary*1.10,2)
  - from employees
  - where department_id=10;
- wydrukuj nazwiska i imiona wszystkich pracowników zatrudnionych od 1 stycznia 2000 roku
  - alter session set nls_date_format='YYYY-MM-DD';
  - select last_name, first_name, hire_date
  - from employees
  - where hire_date >='01-Jan-2000'
- wydrukuj nazwiska i imiona wszystkich pracowników przekonwertowane do dużych liter.
  - SELECT upper(last_name), upper(first_name)
  - FROM employees;
- wydrukuj nazwiska, imiona, daty zatrudnienia oraz pensje wszystkich pracowników. Daty zatrudnienia powinny zawierać NAZWĘ miesiąca, np. 21-JANUARY-1999
  - select last_name, first_name, hire_date, salary
  - from employees;
- wydrukuj unikalną listę dni tygodnia, w które byli zatrudniani poszczególni pracownicy z departamentu 20
  - select distinct to_char(hire_date,'D')
  - from employees
  - where department_id=20;
- wydrukuj nazwiska, imiona oraz inicjały wszystkich pracowników z departamentu 10. Inicjały skonwertuj do małych liter
  - SELECT upper(last_name), upper(first_name), lower(concat(substr(first_name,1,1),substr(last_name,1,1)))
  - FROM employees
  - where department_id=10;
- zbuduj raport zawierający nast. dane wszystkich pracowników: numer departamentu, nazwiska i imiona skonwertowane do dużych liter oraz pensje powiększone o 25 procent i zaokrąglone do dwóch miejsc po przecinku
  - select department_id, last_name, upper(last_name), upper(first_name), round(salary*1.25,2)
  - from employees;
- zbuduj raport zawierający nazwiska wszystkich pracowników w nazwiskach każde wystąpienie litery K zamień na A
  - select last_name, replace(last_name, 'K','A')
  - from employees;
- select count(*)
  - from departments
- select count(distinct commission_pct)
  - from employees;
- select sum(salary)
  - from employees;
- select sum(distinct salary)
  - from employees;
- select avg(distinct salary*12)
  - from employees;
- select sum(all salary)
  - from employees;
- select max(salary + nvl(commission_pct,0)*commission_pct), min(distinct salary*12)
  - from employees;
- SELECT avg(salary)
  - FROM employees
  - GROUP BY department_id;
- SELECT max(salary)
  - FROM employees
  - GROUP BY job_id
- SELECT job_id, avg(salary)
  - FROM employees
  - GROUP BY job_id
- SELECT job_id, department_id, avg(salary)
  - FROM employees
  - GROUP BY job_id, department_id

- zbuduj raport zawierający numer departamentu oraz maks.pensję pracowników w każdym z tych departamentów
  - select department_id, max(salary)
  - from employees
  - group by department_id
- zbuduj raport zawierający stanowisko pracy oraz liczbę pracowników na każdym z owych stanowisk
  - SELECT job_id, count(salary)
  - FROM employees
  - GROUP BY job_id
- który manager ma więcej, niż dwóch podwładnych ?
  - SELECT manager_id, count(manager_id)
  - from employees
  - having count(manager_id)>2
  - group by manager_id
- jakie są najwcześniejsze daty zatrudnienia w poszczególnych departamentach ?
  - SELECT department_id, min(hire_date)
  - from employees
  - group by department_id
- zbuduj raport zawierający nr departamentu, stanowisko pracy oraz maks. pensję pracowników w ramach departamentu i stanowiska pracy
  - select department_id, job_title, max(salary)
  - from employees, jobs
  - WHERE jobs.job_id=employees.job_id
  - group by department_id, job_title
- który departament ma średnią pensję niższą, niż 4000 ?
  - select department_id, avg(salary)
  - from employees
  - having avg(salary)>4000
  - group by department_id
- Napisz polecenie, które zwracać będzie nazwy departamentów, liczby zatrudnionych w nich pracowników oraz ich maks. i min. pensje
  - select department_name, count(*), min(salary), max(salary)
  - from employees, departments
  - WHERE departments.department_id=employees.department_id
  - group by department_name
- zbuduj raport zawierający nazwy departamentów, nazwiska, imiona oraz roczne pensje pracujących w nich pracownikach. Wyłącz z raportu departamenty o numerach większych, niż 80
  - SELECT department_name, last_name, first_name, salary*12
  - FROM departments, employees
  - WHERE departments.department_id = employees.department_id
  - AND departments.department_id<=80
- Zbuduj raport, który będzie zawierał nazwę departamentu oraz średnią pensję wszystkich zatrudnionych w nim pracowników. Wyłącz z raportu departamenty zatrudniające mniej, niż 2 pracowników
  - select departments.department_id, avg(salary), count(*)
  - from departments, employees
  - WHERE departments.department_id = employees.department_id
  - having count(*)>1
  - group by departments.department_id
- Napisz polecenie SELECT odczytujące miasto z tablicy LOCATIONS oraz liczbę departamentów w danym mieście
  - select city, count(department_id)
  - from departments, locations
  - where departments.location_id= locations.location_id
  - group by city
- Napisz polecenie, które zwracać będzie nazwy miast, departamentów w nich zlokalizowanych oraz nazwiska i imiona wszystkich pracowników w poszcz. departamentach. Wyłącz z zestawienia departamenty o numerach 10, 20 oraz 30
  - select city, department_name, last_name, first_name, departments.department_id
  - from departments, locations, employees
  - where departments.location_id= locations.location_id
  - and employees.department_id=departments.department_id
  - and departments.department_id not in (10,20,30)
- Napisz polecenie, które zwracać będzie nazwy departamentów, liczby zatrudnionych w nich pracowników oraz ich maks. i min. pensje
  - select department_name, count(*), min(salary), max(salary)
  - from employees, departments
  - WHERE departments.department_id=employees.department_id
  - group by department_name

- Napisz polecenie, które zwracać będzie stanowiska pracy oraz liczby miast z departamentami, w których pracują osoby na tych stanowiskach pracy
  - select job_title, department_name, count(city)
  - from departments, locations, employees, jobs
  - where departments.location_id= locations.location_id
  - and employees.department_id=departments.department_id
  - and jobs.job_id=employees.job_id
  - group by job_title, department_name
- Napisz polecenie odczytujące stanowiska pracy oraz liczbę departamentów, w których owe stanowiska funkcjonują
  - select job_title, count(department_name)
  - from departments, locations, employees, jobs
  - where departments.location_id= locations.location_id
  - and employees.department_id=departments.department_id
  - and jobs.job_id=employees.job_id
  - group by job_title
- przesuń wszystkich pracowników z departamentu 10 do 20
  - UPDATE employees_copy
  - SET department_id= 20
  - WHERE department_id = 10;
- W kopii tabeli employees, zwiększ pracownikom pensję o 10%
  - UPDATE employees_copy
  - SET salary = salary*1.1
- przy pomocy pow. polecenia, utwórz kopię tabeli EMPLOYEES
  - CREATE TABLE employees_copy
  - AS SELECT *
  - FROM employees;