Test study guide

Simple select statements

ORDER BY

Ascending by default

Can also sort descending by adding DESC or DESCENDING after the list of expressions

You can use aliases from the SELECT list in the ORDER BY clause

WHERE

Understand how to evaluate WHERE clause

LIKE - % replaces zero or more characters; _ replaces one character

BETWEEN is inclusive. So, if you have BETWEEN 1 AND 10, rows with 1 and rows with 10 are included in the results

IN – provide a comma delimited of items

NOT – negates the predicate

When working with strings, understand that 'Big' is greater than 'B'

Operators include =, <, >, <=, >=, <>, !=

You cannot compare anything to NULL, so you must use the ISNULL or COALESCE function to change NULL into something else such as an empty string or another value

OR – at least one expression must return true

AND – both expressions must return true

Use parenthesis to enforce logic when using AND and OR, similar to working a math problem

DISTINCT

Using DISTINCT will eliminate duplicate rows

SELECT DISTINCT ID, Name FROM People;

TOP

Top is used to return a certain number or percent of rows

SELECT TOP(10) ID, Name FROM People;

JOINING

INNER JOIN

The word “inner” is optional

INNER JOIN is part of the FROM clause

Table1 INNER JOIN Table2 ON Table1.ForeignKey = Table2.PrimaryKey

Keys may often have the same name, but often do not

To continue adding tables, continuing adding INNER JOINs

Only rows that that match on both sides of the join will be returned

LEFT OUTER JOIN

The word “outer” is optional

Use LEFT OUTER JOIN when you want all rows returned from the left table even if there is not a match on the right

Columns from the right table will return NULL for rows that don’t match

Once you use LEFT OUTER JOIN, continue using LEFT OUTER JOIN for other table joins

Queries that combine rows from two queries

The two queries must have the same number of columns.

A column must be of a compatible data type between the two queries.

The first query will provide the column names

If you want to add an ORDER BY, add it at the end

UNION

All the rows from two queries are return in one result set

Here is an example:

```
SELECT ID, Name
FROM People
UNION
SELECT BizID, FullName
FROM Customers
```

Duplicate rows are eliminated

UNION ALL

It works like UNION, but will return duplicate rows if there are any

EXCEPT

Returns any rows in the first query that are missing from the second query

This is useful for comparing the results of two queries

Example:

Table1

1, Sam

2, Kellyn

3, Pat

Table2

1, Sam

2, Grace

3, Thomas

```
SELECT ID, Name
FROM Table1
EXCEPT
SELECT ID, Name
FROM Table2
```

Results:

2, Kellyn

3, Pat

INTERSECT

Returns the rows that are included in both queries

Results:

1, Sam

Functions

Understand how to evaluate common functions. Functions can be applied to columns, expressions, variables, or hard-coded values.

GETDATE() – returns the current date and time

YEAR(date)

YEAR('1/1/2018') returns 2018

MONTH(date)

MONTH('1/1/2018') returns 1

ISNULL – replace a NULL value

ISNULL(Color,’no color’) – returns ‘no color’ for any rows where the color is null

COALESCE – works like ISNULL to replace a NULL value, but you can include multiple replacement values

COALESCE(Color, Size, ‘n/a’) – If both the color and size are NULL, then it returns ‘n/a’

CONCAT – combine columns or strings. It automatically takes care of NULL values and converting data types

Table1(ID INT, Name VARCHAR(30))

1, Sam

2, Kellyn

3, Pat

4, NULL

If you wanted to combine the ID and name you have two possible ways to do it.

This example will not work because INT and VARCHAR are not compatible. Also, row 4 will return null.

SELECT ID + ‘-‘, + Name FROM Table1

This example will work

SELECT CONCAT(ID,Name) FROM Table1

1-Sam

2-Kellyn

3-Pat

4-

CHARINDEX

Find a string in another string. The position is returned.

Find the letter a

SELECT Name, CHARINDEX(‘a’,Name) FROM Table1

Sam, 2

Kellyn, 0

Pat, 2

NULL,0

DATEADD

Add time to a date

DATEADD(day,3,’1/1/2018’) returns ‘1/4/2018’

DATEADD(year,1,’1/1/2018’) returns ‘1/1/2019’

You can add any time unit, even microseconds

To subtract, just use a negative number

DATEDIFF

Find the difference between two dates

DATEDIFF(day,’1/1/2018’,’1/3/2018’) returns 2

You can find the difference using any time unit

Supply the earliest date first

LEFT

Return a number of characters from the left side of a string

LEFT(‘abcdefg’,3) returns ‘abc’

RIGHT

Return a number of characters form the right side of a string

RIGHT(‘abcdefg’,3) returns ‘efg’

REVERSE

Reverse a string.

REVERSE(‘abcdefg’) returns ‘gfedcba’

CAST

Change the data type of a value. This is often used when comparing or combining numbers and strings

This returns an error because you can’t compare an integer to a string

SELECT * FROM Table1 WHERE ID =Name

This will work, but may not return any results

SELECT * FROM Table1 WHERE CAST(ID AS VARCHAR(10)) = Name

CONVERT

Also changes the data type, but you can also supply a style to format dates when converting to strings

Table3(ID int, TheDate date)

1,‘2018-01-01’

2,’2018-02-01’

3,’2018-03-01’

SELECT ID, TheDate, CONVERT(varchar(10), TheDate,5) FROM Table3

1,‘2018-01-01’, ’01-01-18’

2,’2018-02-01’,’02-01-18’

3,’2018-03-01’,’03-01-18’

Aggregate

Aggregates are used to group and summarize data.

Functions

The most common are

SUM, COUNT, AVG, MIN, MAX

If you want to count the number of rows, you can use COUNT(*)

NULL values are ignored

GROUP BY

If you want to include any columns in the SELECT or ORDER BY that are not aggregated, they must be in the GROUP BY clause. You will get one row per unique group.

```
SELECT COUNT(*), CustomerID
FROM Sales
GROUP BY CustomerID
```

One row for each CustomerID will be returned with the count of the rows for that CustomerID

Make sure that the exact expression is in the GROUP BY. For example, if YEAR(OrderDate) is in the SELECT list, then make sure YEAR(OrderDate) is in the list.

HAVING

Use having to filter groups. If you want to filter on a value, use the WHERE clause.

```
SELECT COUNT(*), SUM(Col2), Col1
FROM Table
WHERE Col1 =7
GROUP BY Col1
HAVING SUM(Col2) > 1000;
```

Data types

Character data types

NCHAR

NVARCHAR

CHAR

VARCHAR

NVARCHAR(MAX)

VARCHAR(MAX)

Numeric data types

INT

BIGINT

SMALLINT

TINYINT

DateTime data types

DATE

TIME

DATETIME2

Given the requirements of a table, select the correct data types

Constraints

Primary Key

Foreign key

Unique constraint (the column must be unique, and can have one NULL row)

Default (a value that is used if one is not supplied when inserting a row)

Check constraint (a range of values for a column)

Computed column

Table creation

Given a set of requirements for a table, choose the correct definition.

Temp tables begin with a # (local) or ## (global).

Other objects

Views – a saved query. Used to simplify writing queries or control security

User Defined functions –

Scalar – returns one value

Table valued functions – returns tabular data. Can join to other tables with CROSS APPLY or OUTER APPLY

Stored procedures

Stored procedures are the workhorse of SQL Server

They can do work such as CRUD operations (create, insert, update, delete). Even create a database! They are like little programs that can return datasets

You can use IF statements, and WHILE loops