DDL

TABLE	COLUMN
Creation	
CREATE TABLE table_name	ALTER TABLE table_name
column_name1, data_type(size), column_name2 data_type(size), column_name3 data_type(size),	ADD column_name datatype
Renaming	
RENAME old_table_name	ALTER TABLE table_name
TO new_table_name;	RENAME old_column_name TO new__column_name;
Deleting	
DROP TABLE Table_Name	ALTER TABLE Table_Name DROP COLUMN column_name;

DML

ADDING ROWS	DELETING ROWS
INSERT INTO table_name (column1, column2, column3, ...)	DELETE FROM table_name
VALUES (value1, value2, value3, ...);	WHERE condition (*eg , column2 = value2,*;
UPDATING ROWS	
UPDATE table_name SET column1 = value1	, column2 = value2, ... WHERE condition;

DQL

Wildcard Pattern Matching		
SELECT column1,	column2,	FROM table_name
WHERE column2 LIKE pattern;	'_r%' Finds any values that have "r" in the second position	'a%' Finds any values that start with "a".
	'a%o' Finds any values that start with "a" and ends with "o"	'%or%' Finds any values that have "or" in any position.

Sorting

DQL (cont)

Sorts the results of a column alphabetically or numerically, ascending by default	SELECT name_of_column1, name_of_column2 FROM name_of_table WHERE condition1 OR condition2... ASC/DESC;
Contains	
SELECT * FROM TableName	WHERE Country IN ('val1', 'val2', 'val3');
CURSOR	
A database cursor is an object that enables traversal over the rows of a result set. It allows you to process individual row returned by a query	
DECLARE all variables you’ll need	
DECLARE … CURSOR FOR SELECT query, where you’ll declare a cursor and also define the query related to (populating) that cursor	DECLARE cursor_name CURSOR FOR select_statement;
OPEN the cursor and FETCH NEXT from the cursor	OPEN cursor_name;....FETCH NEXT FROM cursor INTO variable_list;
In the WHILE loop you’ll test the @@FETCH_STATUS variable (WHILE @@FETCH_STATUS = 0). If the condition holds, you’ll enter the loop BEGIN … END block and perform statements inside that block	WHILE @@FETCH_STATUS = 0 BEGIN FETCH NEXT FROM cursor_name; END;
CLOSE the cursor and DEALLOCATE it.	CLOSE cursor_name; DEALLOCATE cursor_name;
CASE	

DQL (cont)

Goes through conditions and returns the value corresponding to the first true condition (like an if-then-else statement)	CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 WHEN conditionN THEN resultN ELSE result END;

DDL Examples

TABLE	COLUMN
Creation	
CREATE TABLE friends	ALTER TABLE Friends D
name varchar(100),	ADD id int;
age int);	
Renaming	
RENAME friends	ALTER TABLE fam
TO fam;	RENAME name TO first_name;
Deleting	
DROP TABLE Fam;	ALTER TABLE Fam DROP COLUMN age;

DML Example

ADDING ROWS	DELETING ROWS
INSERT INTO fam (id, name, age)	DELETE FROM fam
VALUES (1, 'Ross', 31);	WHERE condition (id = 1);
UPDATING ROWS	
UPDATE fam SET name = 'Rachel Geller'	WHERE id = 2;

DQL example

Wildcard Pattern Matching		
SELECT column1,	column2,	FROM table_name
WHERE column2 LIKE pattern;	'_r%' Finds any values that have "r" in the second position	'a%' Finds any values that start with "a".
	'a%o' Finds any values that start with "a" and ends with "o"	'%or%' Finds any values that have "or" in any position.
Order By		

DQL example (cont)

	SELECT name_of_column1, name_of_column2 FROM name_of_table WHERE condition1 OR condition2... ASC/DESC;
Contains	
SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'UK');	SELECT * FROM Customers WHERE Country NOT IN ('Germany', 'France', 'UK');
CURSOR	
Allows us to update one row at a time or perform an administrative process such as SQL Server database backups in a sequential manner.	
DECLARE all variables we need	DECLARE @product_name VARCHAR(MAX), @list_price DECIMAL;
DECLARE … CURSOR FOR SELECT naming our cursor and the query to find the values it will contain	DECLARE cursor_product CURSOR FOR SELECT product_name, list_price FROM production.products;
OPEN the cursor and FETCH NEXT from the cursor	OPEN cursor_product; FETCH NEXT FROM cursor_product INTO @product_name, @list_price;

DQL example (cont)

The WHILE loop to test the weather our condition returned values, when 0 (meaning rows were returned), we fetch the specified values	WHILE @@FETCH_STATUS = 0 BEGIN PRINT @product_name + CAST(@list_price AS varchar); FETCH NEXT FROM cursor_product INTO @product_name, @list_price; END;
CLOSE the cursor and DEALLOCATE it.	CLOSE cursor_product; DEALLOCATE cursor_product;
Case	
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 CustomerName, City, Country FROM Customers ORDER BY (CASE WHEN City IS NULL THEN Country ELSE City END);