



# Relational Database Management System, DML

---

.NET

*Software used to maintain relational databases is a Relational Database Management System (RDBMS). Many relational database systems use Structured Query Language (SQL) for querying and maintaining the database.*

[HTTPS://DOCS.MICROSOFT.COM/EN-US/SQL/T-SQL/LANGUAGE-REFERENCE?](https://docs.microsoft.com/en-us/sql/t-sql/language-reference?)

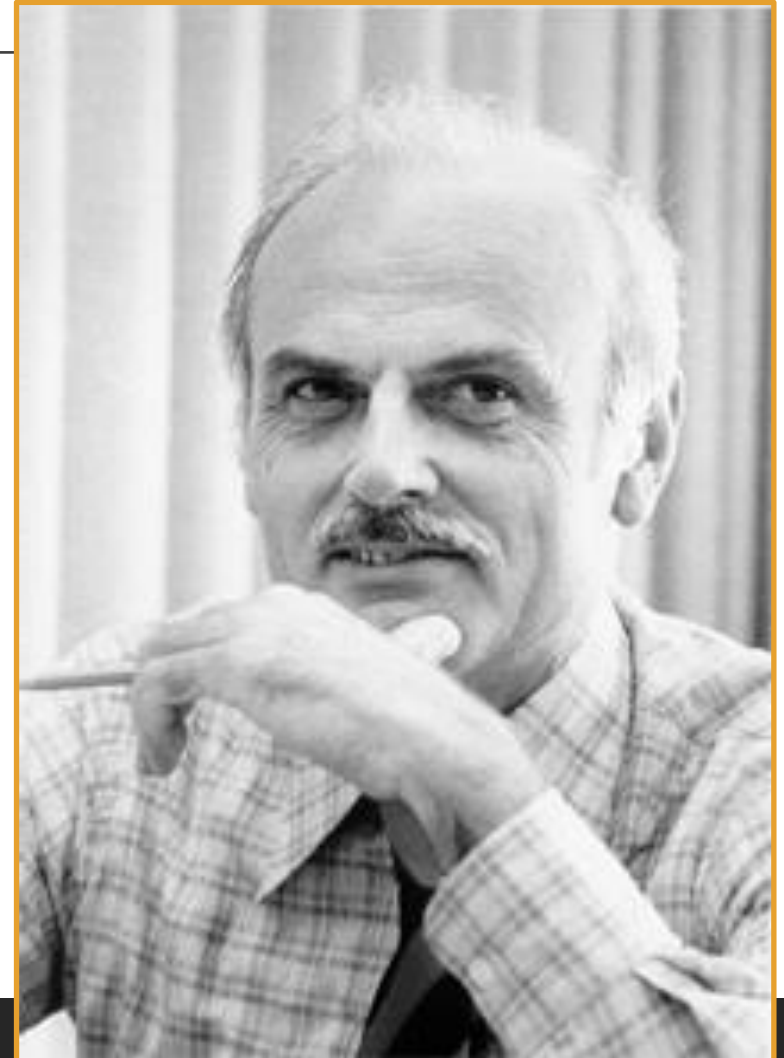
# (RDBMS) Relational Database Management System – History

[https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database)

---

Relational databases are based on the relational model of data, as proposed by E. F. Codd in 1970.

Edgar Frank "Ted" Codd (August 23, 1923 – April 18, 2003) was a British computer scientist and winner of the 1981 Turing Award.



# SQL (Structured Query Language)

<https://en.wikipedia.org/wiki/SQL>

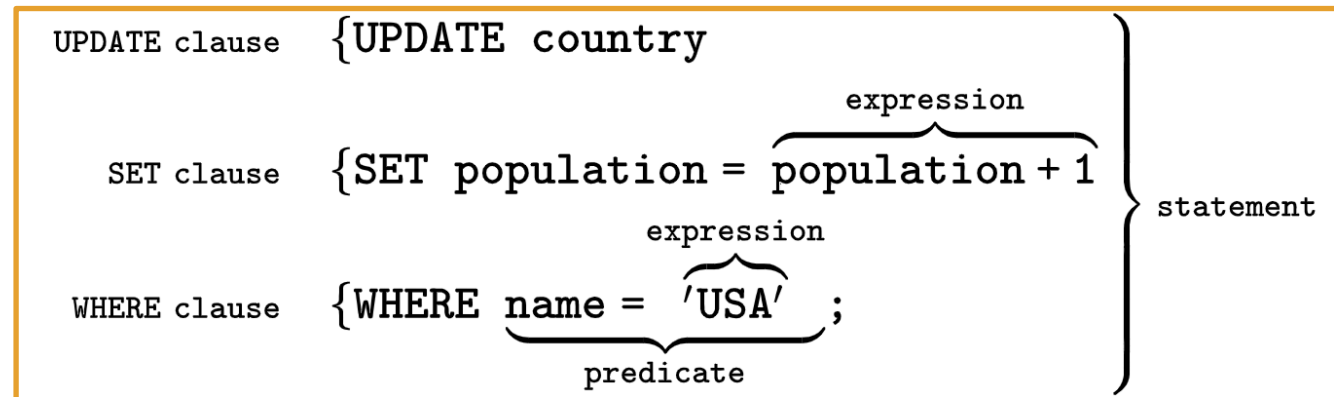
---

SQL was originally based upon relational algebra and tuple relational calculus. SQL is a declarative language. We say what data we want, not how to get it. We cannot manage how SQL obtains the data.

The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control.

SQL consists of two main types of statements:

- Data Definition Language (DDL)
- Data Manipulation Language (DML).



# Microsoft and T-SQL

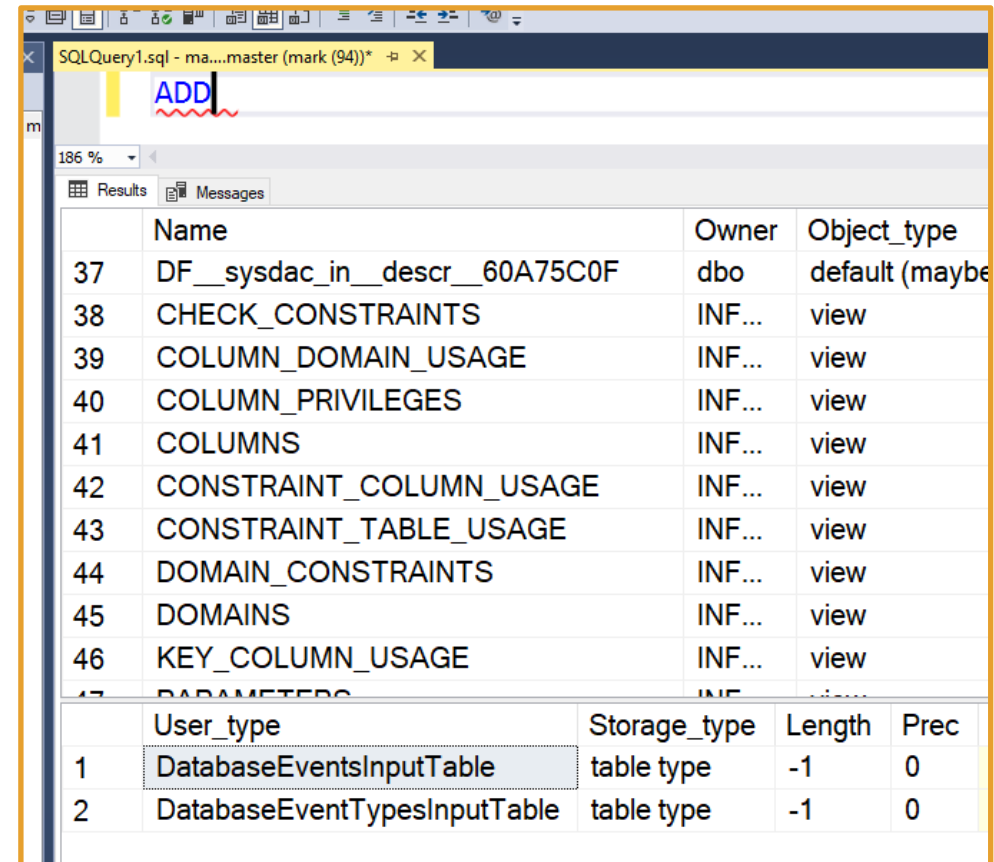
<https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15#tools-that-use-t-sql>

T-SQL is central to using Microsoft SQL products and services. All tools that communicate with a SQL database send T-SQL commands. SQL works on top of T-SQL.

Some of the Microsoft tools that issue T-SQL commands are:

- SQL Server Management Studio (SSMS)
- SQL Server Data Tools (SSDT)
- Azure Data Studio

\*You can type a T-SQL keyword in the SSMS Query Editor window and press F1 to get data about any T-SQL Keyword.

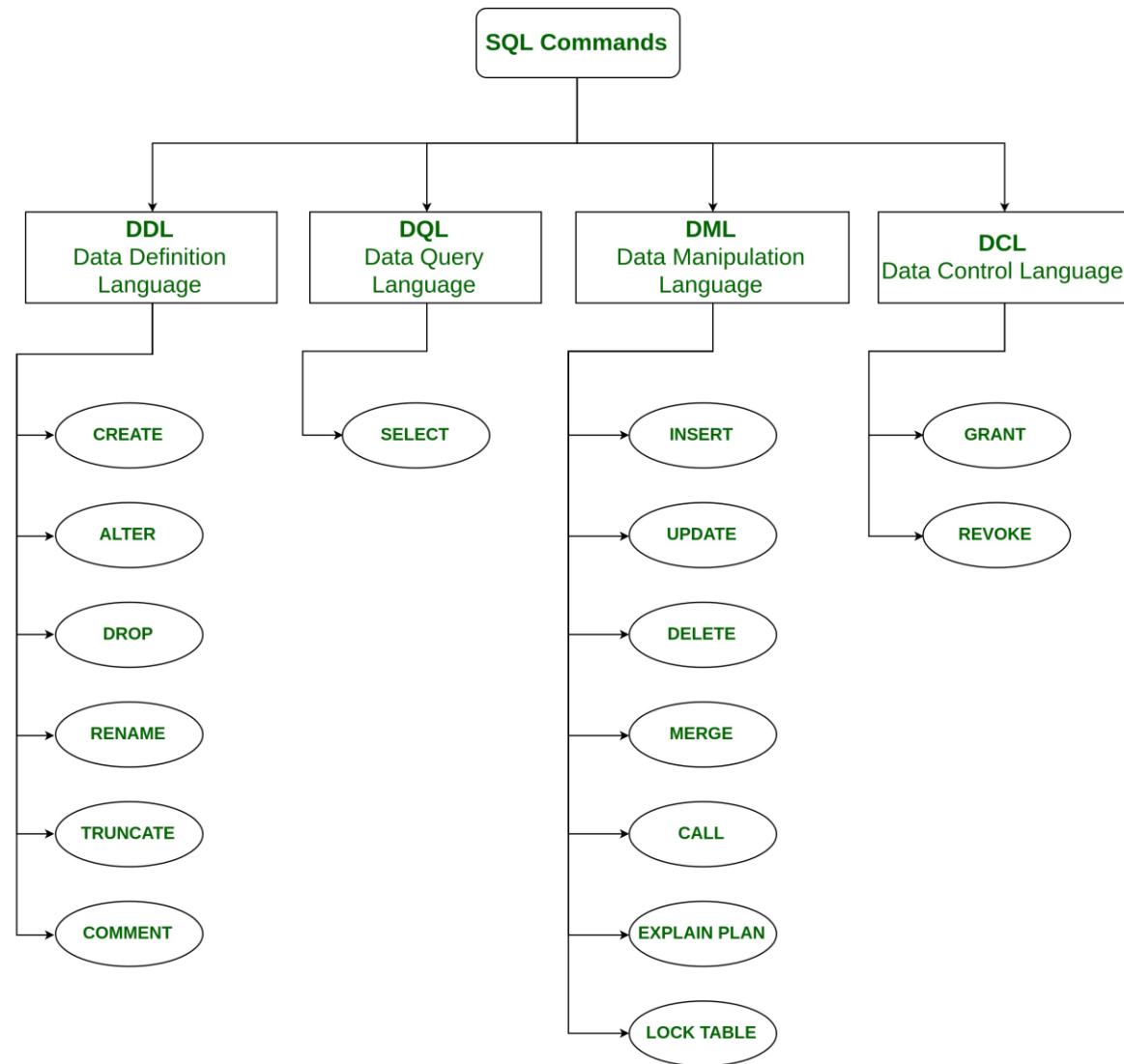


The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the 'ADD' keyword in the Query Editor. The bottom pane, titled 'Results', shows a table of search results for the keyword 'ADD'. The table has columns for 'Name', 'Owner', 'Object\_type', 'User\_type', 'Storage\_type', 'Length', and 'Prec'. The results are as follows:

	Name	Owner	Object_type	User_type	Storage_type	Length	Prec
37	DF__sysdac_in__descr__60A75C0F	dbo	default (maybe)				
38	CHECK_CONSTRAINTS	INF...	view				
39	COLUMN_DOMAIN_USAGE	INF...	view				
40	COLUMN_PRIVILEGES	INF...	view				
41	COLUMNS	INF...	view				
42	CONSTRAINT_COLUMN_USAGE	INF...	view				
43	CONSTRAINT_TABLE_USAGE	INF...	view				
44	DOMAIN_CONSTRAINTS	INF...	view				
45	DOMAINS	INF...	view				
46	KEY_COLUMN_USAGE	INF...	view				
47	PARAMETERS	INF...	view				
1	DatabaseEventsInputTable			table type	table type	-1	0
2	DatabaseEventTypesInputTable			table type	table type	-1	0



## Types of SQL Commands



# Data Manipulation Language

<https://docs.microsoft.com/en-us/sql/t-sql/statements/statements?view=sql-server-ver15#data-manipulation-language>

---

*Data Manipulation Language (DML)* is used to query a Database to retrieve and modify data from an SQL Database. Use *Data Manipulation Language* statements to **INSERT**, **UPDATE**, **DELETE**, **TRUNCATE**, and query the rows in the database.

- **INSERT** – Adds one or more rows to a table or a view.
- **DELETE** – Removes one or more rows from a table or view in SQL Server.
- **UPDATE** – Changes existing data in a table.
- **TRUNCATE** - Removes all rows from a table without deleting the table. Similar to the **DELETE** statement with no **WHERE** clause, but faster.

# SQL - SELECT, FROM, WHERE

<https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15#logical-processing-order-of-the-select-statement>

---

The three most common keywords in SQL Queries are **SELECT**, **FROM**, and **WHERE**.

- **SELECT** - Retrieves rows from the database
- **FROM** – Usually required on the **SELECT** statement. Specifies from which table(s) the results will come.
- **WHERE** - Specifies the search conditions for the rows returned by the query.

```
SELECT first_name, last_name FROM  
student_details  
WHERE id = 100;
```



# SQL – Queries

[https://www.ibm.com/support/knowledgecenter/SSGU8G\\_12.1.0/com.ibm.dbdk.doc/ids\\_dbdk\\_028.htm](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.dbdk.doc/ids_dbdk_028.htm)

---

SQL Queries have a specific syntax and structure.

1. Start the query with the **SELECT** keyword,
2. State the names of the columns you want in the result,
3. State **FROM** which table to get the data,
4. Followed by conditions.

This query returns the TerritoryID and Name columns from the Sales.SalesDirectory table and orders the result by TerritoryID. Default order is ascending.

```
SELECT TerritoryID, Name  
FROM Sales.SalesTerritory  
ORDER BY TerritoryID ;
```

# SQL - Queries

---

Queries are used to **INSERT**, retrieve, modify, and **DELETE** data in a SQL Database.

```
SELECT TerritoryID, Name
FROM Sales.SalesTerritory
ORDER BY TerritoryID ;
```

```
SELECT e.BusinessEntityID, d.Name AS Department
FROM HumanResources.Employee AS e
CROSS JOIN HumanResources.Department AS d
ORDER BY e.BusinessEntityID, d.Name ;
```

```
SELECT DepartmentNumber,
       DepartmentName,
       ManagerID,
       ParentDepartmentNumber
FROM DEPARTMENT
FOR SYSTEM_TIME AS OF '2014-01-01'
WHERE ManagerID = 5;
```

# Query Examples

---

This query **SELECT**s the EmployeeKey and LastName columns from the DimEmployee table, but only if the last name is “Smith”

```
SELECT EmployeeKey, LastName  
FROM DimEmployee  
WHERE LastName = 'Smith' ;
```

This query **SELECT**s all the columns from the Production.Product table and orders the result by the Name column in ascending order.

```
SELECT *  
FROM Production.Product  
ORDER BY Name ASC;
```

# Query Examples – Aggregate Functions

---

This query returns the amount of items in the Quantity column from the table OrderDetails.

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

This query returns the average of all the numbers in the Price column from the Products table.

```
SELECT AVG(Price)
FROM Products;
```

# Query Examples – Aggregate Functions

---

This query returns the CustomerID and Country columns from the Customers table. It will GROUP all the like countries and tell how many customers from each country there are.

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

This query returns the CustomerID and Country columns from the Customers table. It will GROUP all the like countries and tell how many customers from each country there are if the count total is over 5.

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

# SQL – UPDATE a table

<https://docs.microsoft.com/en-us/sql/t-sql/queries/update-transact-sql?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/sql/t-sql/queries/update-transact-sql?view=sql-server-ver15#a-using-a-simple-update-statement>

---

**UPDATE** changes existing data in a table or view. There are many variations to **UPDATE**.

```
USE AdventureWorks2012;
GO
UPDATE Sales.SalesPerson
SET Bonus = 6000, CommissionPct = .10, SalesQuota = NULL;
GO
```

*UPDATE* multiple rows simultaneously.

```
--UPDATE table_name
--SET column1 = value1, column2 = value2, ...
--WHERE condition;
UPDATE Employee
SET TITLE = 'Tech Lead'
Where FirstName = 'Nick';
```

```
USE AdventureWorks2012;
GO
UPDATE Person.Address
SET ModifiedDate = GETDATE();
```



# SQL – INSERT into a table

<https://docs.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15#BasicSyntax>  
<https://docs.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15#BasicSyntax>

---

**INSERT** adds one or more rows to a table or a view in SQL Server.

```
--INSERT INTO table_name (column1, column2, column3, ...)
--VALUES (value1, value2, value3, ...);
SELECT * FROM Employee;

INSERT INTO Employee (LastName, EmployeeId, FirstName, Title)
VALUES ('Escalona', (SELECT MAX(EmployeeId) FROM Employee) + 1, 'Nick', 'Master');

SELECT * FROM Employee
WHERE EmployeeId = 102;
```

# SQL – DELETE from a table

<https://docs.microsoft.com/en-us/sql/t-sql/statements/delete-transact-sql?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/sql/t-sql/statements/delete-transact-sql?view=sql-server-ver15#BasicSyntax>

---

**DELETE** removes one or more rows from a table or view in SQL Server.

```
--DELETE FROM table_name WHERE condition;  
UPDATE Employee  
SET ReportsTo = 1  
Where EmployeeId = 102;  
  
DELETE FROM Employee  
WHERE FirstName LIKE 'N_C%' AND ReportsTo = 102;
```

# IN(), NOT IN()

---

Select \* from Customers where name IN ('Moore', 'Smith', 'Hunter')

# Between

---

Select productid from orders where productid BETWEEN 5 and 10 (BETWEEN is inclusive)

Select productid from orders where productName BETWEEN 'B' and 'L'

# LIKE and ILIKE

---

Select productName from orders where productName LIKE 'Comp%';

\_ means a single char.

% means any number of chars.

Select productPrice from orders where productPrice LIKE '3.%';

# Ordering Data

---

Select \* from orders where productName LIKE 'Comp%' order by price ASC;

...Order by price DESC;



# Distinct

---

Select DISTINCT productName from orders where productName LIKE 'Comp%';

Select DISTINCT productName, productId from orders where productName LIKE 'Comp%'; (this gives distinct productName but the productId can be the same.)

# LIMIT and OFFSET

---

Select productName from orders where productName LIKE 'Comp%' LIMIT 5; (returns only the first 5)

Select productName from orders where productName LIKE 'Comp%' LIMIT 5 OFFSET 6; (returns 5 but starting with the... 7<sup>th</sup> ...so... 7<sup>th</sup>, 8<sup>th</sup>, 9<sup>th</sup>, 10<sup>th</sup>, and 11<sup>th</sup>, )

# Aliases

---

```
SELECT productName AS JimmyJohn FROM orders WHERE productName LIKE 'Comp%' LIMIT 5;
```

# TO\_CHAR()

---

Converts a DATE type to text. Is useful for timestamp formatting

# EXTRACT()

---

Extract a subcomponent of a date value. Like the YEAR, MONTH, DAY, WEEK, QUARTER

Syntax:

```
EXTRACT(YEAR FROM <dateColumn>)
```

# AGE()

---

Calculates and returns the current age of a given timestamp.



# ROUND()

---

# Math functions

---

-- Math functions.

```
SELECT * FROM film;
```

```
SELECT rental_rate/replacement_cost FROM film;
```

```
SELECT ROUND(rental_rate/replacement_cost,2) FROM film;
```

```
SELECT film_id*10 AS "multiplied!" FROM film;
```

# String functions

<https://www.postgresql.org/docs/current/functions-string.html>

---

You can concatenate columns or data to a string for custom output!

# COALESCE()

---

SELECT item,(price - COALESCE(discount,0) AS final FROM tableName

COALESCE() replaces a null value with the given value. It is valuable when doing calculations on a column that might have null values.

# CAST()

---

To cast one data type to another type.

```
SELECT CAST('5' AS INTEGER)
```

```
SELECT CAST(dateColumn AS TIMESTAMP)
```

You can cast an integer as a string to count how many digits it is.

```
SELECT CHAR_LENGTH(CAST(inventory_id AS VARCHAR)) FROM rental
```

.

# NULLIF()

---

NULLIF(arg1, arg2) takes 2 args and returns null if they are equal. Otherwise, returns the first arg.

NULLIF(1,2) returns 1.

NULLIF(2,2) returns null.

# Chinook DB Seeder Code

[https://raw.githubusercontent.com/2002-feb24-net/trainer-code/master/2-sql/Chinook\\_SqlServer.sql](https://raw.githubusercontent.com/2002-feb24-net/trainer-code/master/2-sql/Chinook_SqlServer.sql)

---

Download and set up SQL Server Management Studio – <https://aka.ms/ssmsfullsetup>

SQL Server Express - <https://www.microsoft.com/en-us/download/details.aspx?id=55994>

## Download & Install SQL Server Express

To download SQL Server Express, click on the following link:

<https://www.microsoft.com/en-us/download/details.aspx?id=42299>

# DML Activity

---

- basic exercises in groups of 3 (Chinook database)
- 1. List all customers (full names, customer ID, and country) who are not in the US
- 2. List all customers from brazil
- 3. List all sales agents
- 4. Show a list of all countries in billing addresses on invoices.
- 5. How many invoices were there in 2009, and what was the sales total for that year?
- 6. How many line items were there for invoice #37?
- 7. How many invoices per country?
- 8. Show total sales per country, ordered by highest sales first.



# DML Activity - Answers

```
SELECT * FROM Customer;
```

1. list all customers (full names, customer ID, and country) who are not in the US

```
SELECT CustomerId, FirstName, LastName, Country  
FROM Customer
```

```
WHERE Country != 'USA'; (also acceptable "Country NOT 'USA';")
```

2. list all customers from brazil

3. list all sales agents

```
SELECT *
```

```
FROM Employee
```

```
WHERE Title LIKE '%Sales%Agent%';
```

-- pattern matching with the LIKE operator

-- % - 0 to many of any character

-- [abc] - one of a, b, or c

-- \_ - one of any character

4. show a list of all countries in billing addresses on invoices.

```
SELECT DISTINCT BillingCountry
```

```
FROM Invoice;
```

```
ESC, BillingCountry;
```

# DML Activity - Answers

– **SELECT DISTINCT** means, after you get all the result rows, remove duplicate rows (where ALL column values match)

5. How many invoices were there in 2009,

```
SELECT count(InvoiceDate) FROM Invoice;
```

6. What was the sales total for 2009? (extra challenge: find the invoice count sales total for every year, using one query)

```
SELECT SUM(Total) AS TotalAmount, COUNT(InvoiceId) AS "Invoices in 2009"  
FROM Invoice  
WHERE InvoiceDate BETWEEN '2009-01-01' AND '2010-01-01';  
WHERE YEAR(InvoiceDate) = 2009;
```

```
SELECT YEAR(InvoiceDate) AS Year, SUM(Total) AS TotalAmount, COUNT(InvoiceId) AS Invoices  
FROM Invoice  
GROUP BY YEAR(InvoiceDate);
```

7. how many line items were there for invoice #37?

```
SELECT Count(Quantity) As TotalItems FROM InvoiceLine WHERE InvoiceId = '37';
```

8. How many invoices per country?

```
SELECT BillingCountry, count(BillingCountry) as "Number of Orders" FROM dbo.Invoice GROUP BY BillingCountry
```

9. Show total sales per country, ordered by highest sales first.

```
SELECT BillingCountry, SUM(Total)  
FROM Invoice  
GROUP BY BillingCountry  
ORDER BY SUM(Total);
```