SQL Cheat Sheet

Select

```
SELECT column1, column2, ...
FROM table_name;

SELECT * FROM table_name;
```

SELECT DISTINCT

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

WHERE Clause

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;

SELECT * FROM Customers
WHERE Country='Mexico';
```

AND, OR and NOT Operators

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin'
;

SELECT * FROM Customers
WHERE City='Berlin' OR City='München';

SELECT * FROM Customers
WHERE NOT Country='Germany';
```

ORDER BY Keyword

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;

SELECT * FROM Customers
ORDER BY Country;
```

INSERT INTO

```
INSERT INTO Customers (CustomerName,
City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway'
);
```

NULL Values

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

UPDATE Statement

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

DELETE Statement

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

SELECT TOP Clause

```
SELECT TOP 3 * FROM Customers;

SELECT * FROM Customers
LIMIT 3;

SELECT TOP 50 PERCENT * FROM Customers;
```

(The following SQL statement selects the first 50% of the records from the "Customers" table:)

MIN() and MAX()

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

SQL Cheat Sheet

COUNT(), AVG() and SUM()

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;

SELECT AVG(column_name)
FROM table_name
WHERE condition;

SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

LIKE Operator

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

[charlist] Wildcard

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

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

[!charlist] Wildcard

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

IN Operator

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'U
K');

SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France'
, 'UK');

SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Sup
pliers);
```

BETWEEN

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;

SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

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

SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon
Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;

SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/04/1996# AND #07/09/1996#
;
```

AS

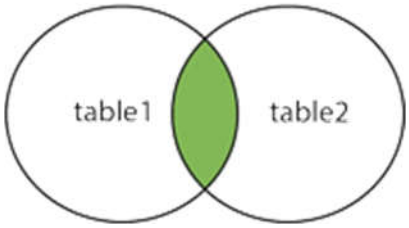
```
SELECT column_name AS alias_name
FROM table_name;

SELECT CustomerID as ID,
CustomerName AS Customer
FROM Customers;

SELECT CustomerName, Address + ', ' +
PostalCode + ' ' + City + ', ' +
Country AS Address
FROM Customers;

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

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;
```

SQL Cheat Sheet

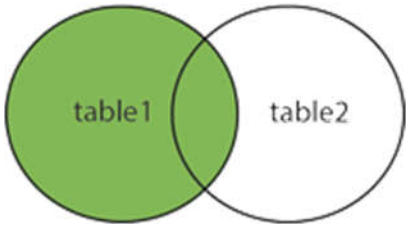
SQL Joins

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

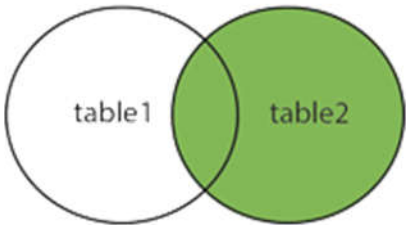
INNER JOIN



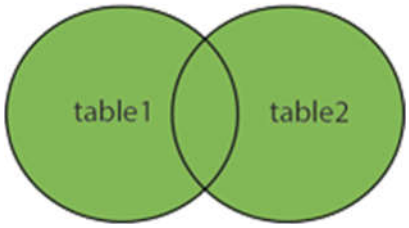
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



join

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table

INNER JOIN

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

JOIN Three Tables

```
SELECT Orders.OrderID,
Customers.CustomerName,
Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.Custom
erID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.Shipper
ID = Shippers.ShipperID);
```

LEFT JOIN

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

RIGHT JOIN

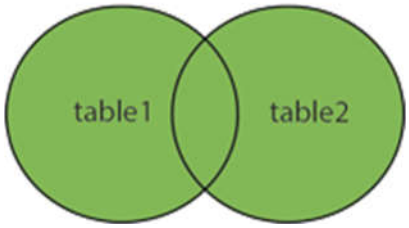
```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID =
Employees.EmployeeID
ORDER BY Orders.OrderID;
```

SQL Cheat Sheet

FULL OUTER JOIN

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

FULL OUTER JOIN



Self JOIN

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

The following SQL statement matches customers that are from the same city:

```
SELECT A.CustomerName AS CustomerName1,
B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

UNION

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;

SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

Each SELECT statement within UNION must have the same number of columns

The columns must also have similar data types

The columns in each SELECT statement must also be in the same order

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

STORED PROCEDURES

```
Create procedure sp_Select_All_PersonContact
As
select Contact.Title, Contact.FirstName,
Contact.LastName
from Person.Contact
//run it
execute sp_Select_All_PersonContact
```

Stored Procedure with an Input Parameter

```
Create procedure sp_Contact_By_Title
@Title nvarchar(8)
As
select Contact.Title, Contact.FirstName,
Contact.LastName
from Person.Contact
where Contact.Title = @Title

execute sp_Contact_By_Title 'Mr.'
```

Stored Procedure with an Output Parameter

```
Create procedure sp_CountContacts_By_Title
@Title nvarchar(8),
@TitleCount int= 0 output
As
select Contact.Title, Contact.FirstName,
Contact.LastName
from Person.Contact
where Contact.Title = @Title
Select @TitleCount = count(*)
from Person.Contact
where Title=@Title
return @TitleCount
//Usage---------------------------------
Declare @return_value int,
@TitleCount int
```

Execute	@return_value=sp_CountContacts_By_Title
@Title='Mr.',	

@TitleCount=@TitleCount	output
Select 'Total Title Count' =@return_value	

SQL Cheat Sheet

Modifying Stored Procedures

```
Alter procedure sp_Select_All_PersonContact
As
select Contact.Title, Contact.FirstName,
Contact.LastName
from Person.Contact
Order by Contact.LastName
```

Displaying the Definition of Stored Procedures

```
sp_helptext 'sp_Select_All_PersonContact'
```

Renaming Stored Procedures

```
sp_rename 'sp_Select_All_PersonContact,
'sp_Select_All_ContactDetails'
```

Deleting Stored Procedures

```
Drop procedure sp_Select_All_ContactDetails
```

FOR XML RAW

```
select Person.Contact.Title,
Person.Contact.FirstName,
Person.Contact.LastNamefrom
Person.Contact
where Person.Contact.Title ='Mr.'
for xml raw
```

FOR XML RAW (Element-centric)

```
select Person.Contact.Title,
Person.Contact.FirstName,
Person.Contact.LastNamefrom
Person.Contact
where Person.Contact.Title ='Mr.'
for xml raw, elements
```

Renaming the row Element

```
select Person.Contact.Title,
Person.Contact.FirstName,
Person.Contact.LastNamefrom
Person.Contact
where Person.Contact.Title ='Mr.'
for xml raw ('PersonDetails'), elements
```