**SQL Quick Reference**

the "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

The "Orders" table:

|  |  |  |
| --- | --- | --- |
| **O\_Id** | **OrderNo** | **P\_Id** |
| 1 | 77895 | 3 |
| 2 | 44678 | 3 |
| 3 | 22456 | 1 |
| 4 | 24562 | 1 |
| 5 | 34764 | 15 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SQL Statement** | **Syntax** | | **Example** |  |
| **AND / OR** | | SELECT column\_name(s) FROM table\_name WHERE condition AND|OR condition | SELECT \* FROM Persons WHERE FirstName='Tove' AND LastName='Svendson'  **o/p:**  **2, svendson,tove,borgvn23,sandnes** | The AND & OR operators are used to filter records based on more than one condition.  The AND operator displays a record if both the first condition and the second condition is true.  The OR operator displays a record if either the first condition or the second condition is true. |
| **ALTER TABLE** | | ALTER TABLE table\_name  ADD column\_name datatype  **(or)**  ALTER TABLE table\_name  DROP COLUMN column\_name  **(or)**  ALTER TABLE table\_name ALTER COLUMN column\_name datatype  **/\*change the data type of a column** | ALTER TABLE Persons ADD DateOfBirth date  (Or)  ALTER TABLE Persons DROP COLUMN DateOfBirth  (Or)  ALTER TABLE Persons ALTER COLUMN DateOfBirth year | The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. |
| **AS (alias)** | | SELECT column\_name AS column\_alias FROM table\_name  (**or)**  SELECT column\_name FROM table\_name  AS table\_alias | SELECT po.OrderID, p.LastName, p.FirstName FROM Persons AS p, Product\_Orders AS po WHERE p.LastName='Hansen' WHERE p.FirstName='Ola' | With SQL, an alias name can be given to a table or to a column. |
| **BETWEEN** | | SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2 | SELECT \* FROM Persons WHERE LastName BETWEEN 'Hansen' AND 'Pettersen'  **o/p: 1, Hansen, Ola, Timoteivn 10, Sandnes** | The BETWEEN operator is used in a WHERE clause to select a range of data between two values.  The values can be numbers, text, or dates. |
| **CREATE DATABASE** | | CREATE DATABASE database\_name | CREATE DATABASE my\_db | The CREATE DATABASE statement is used to create a database. |
| **CREATE TABLE** | | CREATE TABLE table\_name ( column\_name1 data\_type, column\_name2 data\_type, column\_name2 data\_type, ... ) | CREATE TABLE Persons ( P\_Id int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255) ) | The CREATE TABLE statement is used to create a table in a database. |
| **CREATE INDEX** | | CREATE INDEX index\_name ON table\_name (column\_name)  /\**Duplicate values are allowed*  **(Or)**  CREATE UNIQUE INDEX index\_name ON table\_name (column\_name)  /\**Duplicate values are not allowed* | CREATE INDEX PIndex ON Persons (LastName, FirstName)  (or)  CREATE UNIQUE INDEX PIndex ON Persons (LastName) | The CREATE INDEX statement is used to create indexes in tables.  Indexes allow the database application to find data fast; without reading the whole table.  An index can be created in a table to find data more quickly and efficiently |
| **CREATE VIEW** | | CREATE VIEW view\_name AS SELECT column\_name(s) FROM table\_name WHERE condition  **[Read: create, update, and delete a view.]** | CREATE VIEW [Current Product List] AS SELECT ProductID,ProductName FROM Products WHERE Discontinued=No | A view is a virtual table. a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. |
| **DELETE** | | DELETE FROM table\_name WHERE some\_column=some\_value  **(Or)**  DELETE FROM table\_name  (Note: *Deletes the entire table!! without deleting the table)*  (Or)  DELETE \* FROM table\_name  (Note: *Deletes the entire table!! without deleting the table)* | DELETE FROM Persons WHERE LastName='Tjessem' AND FirstName='Jakob' | The DELETE statement is used to delete records(rows) in a table. |
| **DROP DATABASE** | | DROP DATABASE database\_name |  | The DROP DATABASE statement is used to delete a database. |
| **DROP INDEX** | | DROP INDEX table\_name.index\_name (SQL Server) DROP INDEX index\_name ON table\_name (MS Access) DROP INDEX index\_name (DB2/Oracle) ALTER TABLE table\_name DROP INDEX index\_name (MySQL) |  | The DROP INDEX statement is used to delete an index in a table. |
| **DROP TABLE** | | DROP TABLE table\_name |  | The DROP TABLE statement is used to delete a table. |
| **GROUP BY** | | SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name | SELECT Customer,SUM(OrderPrice) FROM Orders GROUP BY Customer | The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns. |
| **HAVING** | | SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name HAVING aggregate\_function(column\_name) operator value | SELECT Customer,SUM(OrderPrice) FROM Orders GROUP BY Customer HAVING SUM(OrderPrice)<2000 | The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. |
| **IN** | | SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1,value2,..) | SELECT \* FROM Persons WHERE LastName IN ('Hansen','Pettersen')  **o/p:**  1, Hansen, Ola, Timoteivn 10, Sandnes  3, Pettersen, Kari ,Storgt 20, Stavanger | The IN operator allows you to specify multiple values in a WHERE clause. |
| **INSERT INTO** | | INSERT INTO table\_name VALUES (value1, value2, value3,....)  ***or***  INSERT INTO table\_name (column1, column2, column3,...) VALUES (value1, value2, value3,....) | INSERT INTO Persons VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')  (or)  NSERT INTO Persons (P\_Id, LastName, FirstName) VALUES (5, 'Tjessem', 'Jakob') | The INSERT INTO statement is used to insert a new row(records) in a table. |
| **INNER JOIN** | | SELECT column\_name(s) FROM table\_name1 INNER JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name | SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons INNER JOIN Orders ON Persons.P\_Id=Orders.P\_Id ORDER BY Persons.LastName | The INNER JOIN keyword return rows when there is at least one match in both tables. |
| **LEFT JOIN** | | SELECT column\_name(s) FROM table\_name1 LEFT JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name | SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons LEFT JOIN Orders ON Persons.P\_Id=Orders.P\_Id ORDER BY Persons.LastName | The LEFT JOIN keyword returns all rows from the left table (table\_name1), even if there are no matches in the right table (table\_name2). In some databases LEFT JOIN is called LEFT OUTER JOIN. |
| **RIGHT JOIN** | | SELECT column\_name(s) FROM table\_name1 RIGHT JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name | SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons RIGHT JOIN Orders ON Persons.P\_Id=Orders.P\_Id ORDER BY Persons.LastName | The RIGHT JOIN keyword Return all rows from the right table (table\_name2), even if there are no matches in the left table (table\_name1). |
| **FULL JOIN** | | SELECT column\_name(s) FROM table\_name1 FULL JOIN table\_name2  ON table\_name1.column\_name=table\_name2.column\_name | ELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons FULL JOIN Orders ON Persons.P\_Id=Orders.P\_Id ORDER BY Persons.LastName | The FULL JOIN keyword return rows when there is a match in one of the tables. |
| **LIKE** | | SELECT column\_name(s) FROM table\_name WHERE column\_name LIKE pattern | SELECT \* FROM Persons WHERE City LIKE 's%'  **(or)**  SELECT \* FROM Persons WHERE City LIKE '%s'  **(or)**  SELECT \* FROM Persons WHERE City NOT LIKE '%tav%' | The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. |
| **ORDER BY** | | SELECT column\_name(s) FROM table\_name ORDER BY column\_name [ASC|DESC] | SELECT \* FROM Persons ORDER BY LastName  **(or)**  SELECT \* FROM Persons ORDER BY LastName DESC | The ORDER BY keyword is used to sort the result-set by a specified column.  The ORDER BY keyword sort the records in ascending order by default.  descending order, you can use the DESC keyword. |
| **SELECT** | | SELECT column\_name(s) FROM table\_name | SELECT LastName,FirstName FROM Persons | The SELECT statement is used to select data from a database. |
| **SELECT \*** | | SELECT \* FROM table\_name | SELECT \* FROM Persons | The asterisk (\*) is a quick way of selecting all columns! |
| **SELECT DISTINCT** | | SELECT DISTINCT column\_name(s) FROM table\_name | SELECT DISTINCT City FROM Persons  o/p:  Sandnes  Stavanger | The DISTINCT keyword can be used to return only distinct (different) values. |
| **SELECT INTO** | | SELECT \* INTO new\_table\_name [IN externaldatabase] FROM old\_table\_name  or  SELECT column\_name(s) INTO new\_table\_name [IN externaldatabase] FROM old\_table\_name | SELECT \* INTO Persons\_Backup FROM Persons  **(or)**  SELECT LastName,FirstName INTO Persons\_Backup FROM Persons  **(or)**  SELECT \* INTO Persons\_Backup IN 'Backup.mdb' FROM Persons | The SELECT INTO statement selects data from one table and inserts it into a different table.  The SELECT INTO statement is most often used to create backup copies of tables.  We can also use the IN clause to copy the table into another database |
| **SELECT TOP** | | SELECT TOP number|percent column\_name(s) FROM table\_name | SELECT \* FROM Persons LIMIT 5  **(or)**  SELECT \* FROM Persons WHERE ROWNUM <=5  **(or)**  SELECT TOP 50 PERCENT \* FROM Persons | The TOP clause is used to specify the number of records to return.  The TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance. |
| **TRUNCATE TABLE** | | TRUNCATE TABLE table\_name |  | What if we only want to delete the data inside the table, and not the table itself?  Then, use the TRUNCATE TABLE statement: |
| **UNION** | | SELECT column\_name(s) FROM table\_name1 UNION SELECT column\_name(s) FROM table\_name2 | SELECT E\_Name FROM Employees\_Norway UNION SELECT E\_Name FROM Employees\_USA | The UNION operator is used to combine the result-set of two or more SELECT statements.  each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.  The column names in the result-set of a UNION are always equal to the column names in the first SELECT statement in the UNION. |
| **UNION ALL** | | SELECT column\_name(s) FROM table\_name1 UNION ALL SELECT column\_name(s) FROM table\_name2 | SELECT E\_Name FROM Employees\_Norway UNION ALL SELECT E\_Name FROM Employees\_USA | The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL. |
| **UPDATE** | | UPDATE table\_name SET column1=value, column2=value,... WHERE some\_column=some\_value | UPDATE Persons SET Address='Nissestien 67', City='Sandnes' WHERE LastName='Tjessem' AND FirstName='Jakob' | The UPDATE statement is used to update existing records in a table. |
| **WHERE** | | SELECT column\_name(s) FROM table\_name WHERE column\_name operator value | SELECT \* FROM Persons WHERE City='Sandnes' | The WHERE clause is used to extract only those records that fulfill a specified criterion. |