Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large OBjects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

(vir: https://www.w3schools.com/MySQL/default.asp)

Numeric Data Types

Data type	Description
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

(vir: https://www.w3schools.com/MySQL/default.asp)

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

(vir: https://www.w3schools.com/MySQL/default.asp)

Ustvaritev

Baze:

CREATE DATABASE imeBaze;

Tabele:

CREATE TABLE Zaposleni (

Employee_ID INT,

First_Name VARCHAR (50),

Last_name VARCHAR (50),

Placa DECIMAL (6, 2)

);

Izbris:

Baze:

DROP DATABASE imeBaze;

Tabele:

DROP TABLE imeTabele;

Kreiramo vrstice:

INSERT INTO Zaposleni

VALUES (2, "Klara", "Novak", 6.7), (3, "Miha", "Jesen", 6.6);

SELECT * FROM Zaposleni;

SELECT:

SELECT * FROM Zaposleni;

SELECT First_Name, Placa;

SELECT DISTINCT First_Name FROM Zaposleni;

MySQL WHERE (pogoj)

SELECT First_Name

FROM Zaposleni

WHERE Placa > 6.0;

SELECT First_Name

FROM Zaposleni

WHERE First_name != "Metka";

SELECT First_Name

FROM Zaposleni

WHERE First_name != "Metka" AND Last_Name != "Novak";

```
SELECT  First_Name
FROM Zaposleni
WHERE First_name != "Metka" OR Last_Name != "Novak";
```

```
SELECT  First_Name
FROM Zaposleni
WHERE NOT First_name = "Metka" AND Last_Name = "Novak";
```

!= ali <>
pomeni da
nista enaka

```
SELECT  First_Name
FROM Zaposleni
WHERE NOT First_name = "Metka" AND NOT Last_Name = "Praprotnik";
```

MySQL ORDER BY (vrstni red)

```
SELECT  First_Name
FROM Zaposleni
ORDER BY First_name;
```

```
SELECT  First_Name, Placa
FROM Zaposleni
ORDER BY Placa;
```

Če želimo, da pada:

```
SELECT  First_Name, Placa
FROM Zaposleni
ORDER BY Placa DESC;
```

MySQL UPDATE (posodobitev)

```
UPDATE Zaposleni
SET Placa = 10
WHERE First_Name = "Kaja";
```

če ne damo pogoja, potem se vse posodobi...

```
SELECT  First_Name, Placa
FROM Zaposleni
ORDER BY Placa DESC;
```

```
UPDATE Zaposleni
SET Placa = 10;
```

```
SELECT  First_Name, Placa
FROM Zaposleni
ORDER BY Placa DESC;
```

MySQL DELETE (Izbris)

DELETE Syntax

```
DELETE FROM Zaposleni
WHERE First_Name = Metka";
```

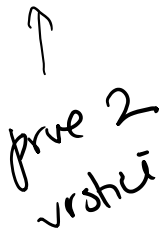
```
SELECT  First_Name, Placa
FROM Zaposleni
```

```
ORDER BY Placa DESC;
```

MySQL LIMIT (postavljenje limite)

LIMIT Syntax

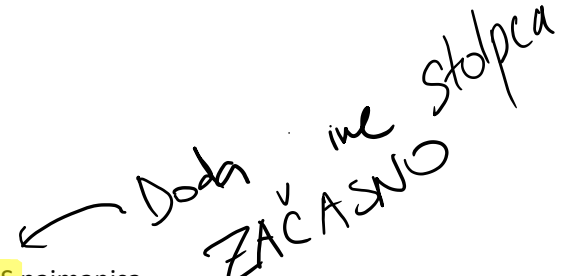
```
SELECT  First_Name, Placa
FROM Zaposleni
ORDER BY Placa DESC
limit 2;
```



MySQL MIN() and MAX() funkcije

MIN() Syntax

```
SELECT  MAX(Placa)
FROM Zaposleni
ORDER BY Placa DESC
limit 5;
```




```
SELECT  MAX(Placa) AS najmanjsa
FROM Zaposleni
ORDER BY Placa DESC
limit 5;
```

COUNT(), AVG() and SUM() funkcije

COUNT() Syntax

SELECT COUNT(Placa) AS stevec

FROM Zaposleni;

povprecije

AVG():

SELECT AVG(Placa) AS povprecna_placa

FROM Zaposleni;

SUM:

SELECT SUM(Placa) AS povprecna_placa

FROM Zaposleni;

MySQL LIKE operator

SELECT First_Name

FROM Zaposleni

WHERE First_Name LIKE '%ka';

Wildcard Characters: (vir: https://www.w3schools.com/MySQL/default.asp)

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt

LIKE Operator	Description
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"

SQL IN operator

SELECT First_Name

FROM Zaposleni

WHERE First_Name IN ("Metka","Nika");

SELECT First_Name

FROM Zaposleni

WHERE First_Name NOT IN ("Metka","Nika");

BETWEEN operator

```
SELECT First_Name
FROM Zaposleni
WHERE placa BETWEEN 7 AND 10;

SELECT First_Name
FROM Zaposleni
WHERE placa NOT BETWEEN 7 AND 10;
```

SQL Aliases

```
SELECT First_Name AS Nickname
FROM Zaposleni;

SELECT First_Name AS [vec nastetih stvari]
FROM Zaposleni;
```

PRIMARY KEY on CREATE TABLE (primarni ključ)

```
CREATE TABLE podjetje(
        id_podjetje INT PRIMARY KEY,
```

```
    ime_podjetja VARCHAR (50)
);

SELECT * FROM Podjetje;
```

Dodamo v že obstoječo tabelo:

```
ALTER TABLE zaposleni
ADD CONSTRAINT
PRIMARY KEY(ID_zaposleni);

SELECT * FROM zaposleni;
```

MySQL FOREIGN KEY (tuji ključ)

```
CREATE TABLE skladisca(
    id_skladisca INT PRIMARY KEY,
    naslov_skladisca VARCHAR (50),
     id_podjetje INT,
     FOREIGN KEY(id_podjetje) REFERENCES podjetje(id_podjetje)
);

SELECT * FROM skladisca;
```

Da odstranimo tuji ključ:

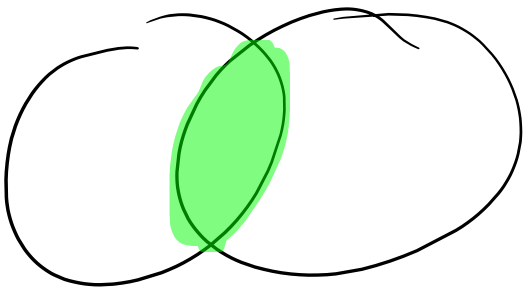
```
ALTER TABLE skladisca
```

```
DROP FOREIGN KEY skladisca_ibfk_1;

SELECT * FROM skladisca;
```

Da dodamo tuji ključ že obstoječi tabeli:

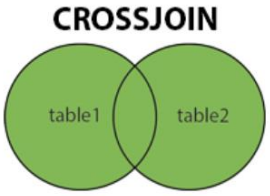
```
ALTER TABLE skladisca
ADD CONSTRAINT tk_id_podjetje
FOREIGN KEY(ID_podjetje) REFERENCES podjetje(ID_podjetje);

SELECT * FROM skladisca;


INSERT INTO podjetje
VALUES (111, "KC"), (112, NULL), (113, "bv"), (114,"NM");

SELECT * FROM podjetje;

INSERT INTO skladisca
VALUES (12, NULL , 111), (13,"AC_S", 112), (14, "bv_S", 113), (15,"NM_S", 114), (16, "BR", NULL);

SELECT * FROM skladisca;
```

INNER JOIN:



```
SELECT *
FROM skladisca
INNER JOIN podjetje
ON skladisca.id_podjetje = podjetje.id_podjetje;
```

SQL CROSS JOIN (združitev)




```
SELECT *
FROM skladisca
CROSS JOIN podjetje;
```

MySQL GROUP BY

Povprecna placa glede na priimek:

```
SELECT AVG(placa), priimek
FROM zaposleni
GROUP BY priimek;
```

Sesševanje plac glede na priimek:

```
SELECT sum(placa), priimek
FROM zaposleni
GROUP BY priimek;
```

MySQL HAVING (POGOJ PRI GROUP BY)

```
SELECT sum(placa), priimek
```

```
FROM zaposleni
GROUP BY priimek
HAVING COUNT(priimek) > 1;
```