



Owner:

[**Azamat**](https://t.me/Azamat_Nishonov) **[Nishonov](https://t.me/sql_portfolio_24)**

SQL Developer

Instructor site: <https://www.w3schools.com/sql>

Content

[SELECT bayonoti 5](#_Toc134621644)

[DISTINCT bayonoti 5](#_Toc134621645)

[WHERE bandi 5](#_Toc134621646)

[AND, OR, NOT 6](#_Toc134621647)

[ORDER BY 6](#_Toc134621648)

[INSERT INTO bayonoti 6](#_Toc134621649)

[NULL qiymatlar 7](#_Toc134621650)

[UPDATE bayonoti 7](#_Toc134621651)

[DELETE bayonoti 8](#_Toc134621652)

[SELECT TOP 8](#_Toc134621653)

[MIN() va MAX() funksiyalari 9](#_Toc134621654)

[COUNT(), AVG() va SUM() funksiyalari 9](#_Toc134621655)

[LIKE operatori 10](#_Toc134621656)

[IN operatori 11](#_Toc134621657)

[BETWEEN operatori 12](#_Toc134621658)

[BETWEEN va IN misol 12](#_Toc134621659)

[Taxalluslar 14](#_Toc134621660)

[Concat 15](#_Toc134621661)

[JOIN 16](#_Toc134621662)

[INNER Join 17](#_Toc134621663)

[Misol 17](#_Toc134621664)

[Misol 17](#_Toc134621665)

[Misol 17](#_Toc134621666)

[SELF Join 18](#_Toc134621667)

[UNION operatori 19](#_Toc134621668)

[Misol 20](#_Toc134621669)

[Misol 20](#_Toc134621670)

[Misol 20](#_Toc134621671)

[Misol 20](#_Toc134621672)

[GROUP BY bayonoti 21](#_Toc134621673)

[HAVING bandi 22](#_Toc134621674)

[EXISTS operatori 24](#_Toc134621675)

[EXISTS sintaksisi 24](#_Toc134621676)

[ANY(some) va ALL operatorlari 25](#_Toc134621677)

[INSERT INTO SELECT bayonoti 28](#_Toc134621678)

[Misol 29](#_Toc134621679)

[CASE ifodasi 29](#_Toc134621680)

[Misol 30](#_Toc134621681)

[COALESCE 31](#_Toc134621682)

[NVL 32](#_Toc134621683)

[Comments 32](#_Toc134621684)

[OPERATORLAR 33](#_Toc134621685)

[Arifmetik operatorlar 33](#_Toc134621686)

[Bitwise operatorlari 33](#_Toc134621687)

[BitAND operatori 34](#_Toc134621688)

[Solishtirish operatorlari 34](#_Toc134621689)

[Mantiqiy operatorlari 34](#_Toc134621690)

[CREATE TABLE. DROP TABLE. 35](#_Toc134621691)

[JADVAL YARATISH VA O`CHIRISH 35](#_Toc134621692)

[Mavjud jadval yordamida yangi jadval yaratish 35](#_Toc134621693)

[ALTER TABLE bayonoti 36](#_Toc134621694)

[CONSTRAINTS (cheklovlar) 38](#_Toc134621695)

[NOT NULL cheklovi 39](#_Toc134621696)

[UNIQUE (UNIKAL) cheklovi 40](#_Toc134621697)

[PRIMARY KEY cheklovi 41](#_Toc134621698)

[FOREIGN KEY cheklovi 42](#_Toc134621699)

[ChECK cheklovi (Tekshiruv) 43](#_Toc134621700)

[DEFAULT cheklovi 44](#_Toc134621701)

[INDEX bayonoti 46](#_Toc134621702)

[Normal indeks yaratish: 47](#_Toc134621703)

[CREATE INDEX sintaksisi 48](#_Toc134621704)

[Funksiyaga asoslangan indeks yaratish: 48](#_Toc134621705)

[Indeks nomini o'zgartirish 49](#_Toc134621706)

[Indeksni o’chirish 49](#_Toc134621707)

[AUTO INCREMENT ⬄ SEQUENCE 49](#_Toc134621708)

[CACHE va NOCACHE farqi 50](#_Toc134621709)

[Nextval 50](#_Toc134621710)

[Sequence ni o’chirish: 51](#_Toc134621711)

[LASTVALUE ni o’zgartirish 51](#_Toc134621712)

[SANA FORMATINI KO’RISH VA O’ZGARTIRISH 51](#_Toc134621713)

[DATES 52](#_Toc134621714)

[sana ma’lumot turlari 52](#_Toc134621715)

[VIEWS 54](#_Toc134621716)

[VIEW ni yangilash 55](#_Toc134621717)

# SELECT bayonoti

SELECT ma’lumotlarni tanlash uchun ishlatiladi.

Jadvaldagi barcha maydonlarni tanlash uchun:

SELECT \* FROM table\_name;

Jadvaldagi ayrim ustunlarni tanlash uchun:

-> SELECT column1, column2, ...  
-> FROM table\_name;

# DISTINCT bayonoti

SELECT DISTINCT turli qiymatlarni qaytarish uchun ishlatiladi.

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

SELECT DISTINCT Country

FROM Customers;

Quyida mijozlarning turli xil mamlakatlari soni ko`rsatilgan:

SELECT COUNT(DISTINCT Country)

FROM Customers;

# WHERE bandi

WHERE 🡪 Shartga munosib filtrlash.

SELECT, UPDATE, DELETE larda ham qo`llaniladi.

WHERE bandidagi operatorlar:= > < >= <= <> !=

SELECT \* FROM table\_name  
WHERE condition;

SELECT \* FROM Customers  
WHERE Country=‘Mexico`;

SELECT \* FROM Customers  
WHERE CustomerID=1;

# AND, OR, NOT

SELECT \* FROM Customers  
WHERE Country=‘Germany’ AND (City=‘Berlin’ OR City=‘Munchen’);

# ORDER BY

ORDER BY Kalit so`z natijalar to`plamini o`sish yoki kamayish tartibida saralash uchun ishlatiladi.

ASC => raqam 🡹, harf a-z. ASC ni yozmasa ham bo`ladi.

SELECT \* FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

Quyida "Mamlakat" va "Mijoz nomi" ustuni bo`yicha tartiblangan. *Country* alifbo tartibida chiqadi, *mamlakat\_nomi* bir xil bo`lgan *CustomerName* lar teskari tartibda chiqadi.

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

# INSERT INTO bayonoti

INSERT INTO jadvalga yangi yozuvlarni kiritish uchun ishlatiladi.

Barcha ustunlarga qiymatlar kiritish:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

Tanlangan ustunlarga qiymatlar kiritish:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES

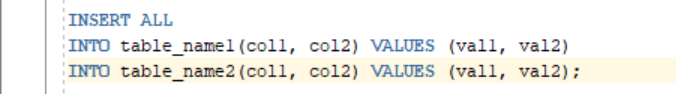
(value01, value02, value03, ...),

(value11, value12, value13, ...);

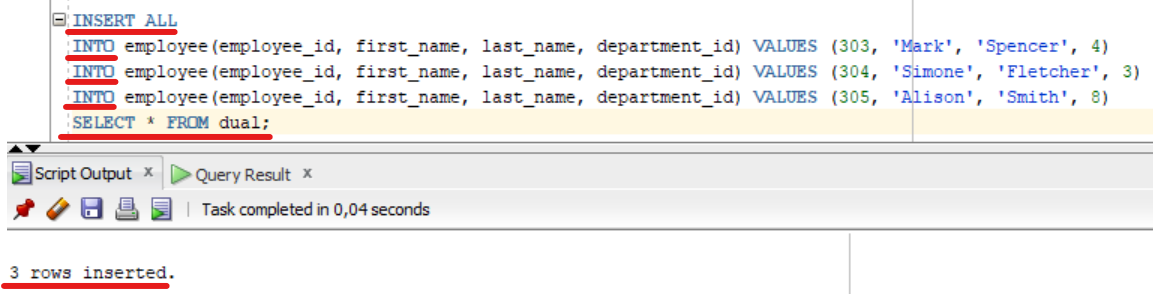
Misol:

INSERT INTO Customers (CustomerName, City, Country)  
VALUES (‘Cardinal’, ’Stavanger’, ’Norway’);

INSERT ALL. Ma’lumotni birdaniga ko`p kiritish uchun shablon:



Misol:



# NULL qiymatlar

NULL qiymatiga ega bo`lgan maydon - yozuv yaratish vaqtida bo`sh qoldirilgan maydondir!

Tekshirishda IS NULL va IS NOT NULL operatorlari ishlatiladi.

SELECT \* FROM table\_name  
WHERE column\_name IS NULL;

SELECT \* FROM table\_name  
WHERE column\_name IS NOT NULL;

# UPDATE bayonoti

UPDATE jadvaldagi mavjud yozuvlarni o`zgartirish uchun ishlatiladi. Yozuvlarni yangilashda ehtiyot bo`ling. Agar siz WHERE bandni o`tkazib yuborsangiz, HAMMA yozuvlar yangilanadi!

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

Misol:

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

# DELETE bayonoti

DELETE jadvaldagi mavjud yozuvlarni o`chirish uchun ishlatiladi. Agar siz WHERE bandni o`tkazib yuborsangiz, jadvaldagi barcha yozuvlar o`chiriladi!

DELETE FROM table\_name

WHERE condition;

Jadvalni o`chirmasdan, faqat barcha qatorlarni o`chirish uchun:

DELETE FROM table\_name;

# SELECT TOP

Ushbu SELECT TOP bandi chiqariladigan yozuvlar sonini cheklaydi.

SELECT column\_name(s)  
FROM table\_nameORDER BY column\_name(s)  
FETCh FIRST number ROWS ONLY;

SELECT \* FROM table\_name  
WHERE ROWNUM < number;

SELECT \* FROM Customers  
FETCh FIRST 50 PERCENT ROWS ONLY;

SELECT \* FROM table\_name  
WHERE condition  
FETCh FIRST 5 ROWS ONLY;

SELECT \* FROM Customers

WHERE ROWNUM < 3;

# MIN() va MAX() funksiyalari

MIN() tanlangan ustunning eng kichik qiymatini qaytaradi.

MAX() eng katta qiymatini qaytaradi.

SELECT MIN(column\_name)  
FROM table\_name;

SELECT MAX(Price)

AS SmallestPrice 🡪 (Taxalluslar bo`limiga qarang.)  
FROM Products;

# COUNT(), AVG() va SUM() funksiyalari

Bular matematik funksiyalardir.

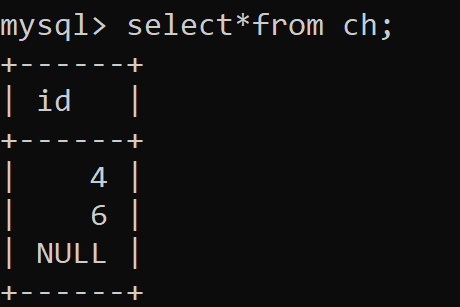
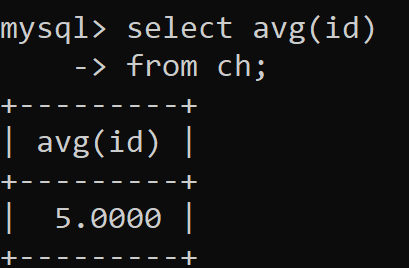
COUNT() qatorlar sonini qaytaradi.

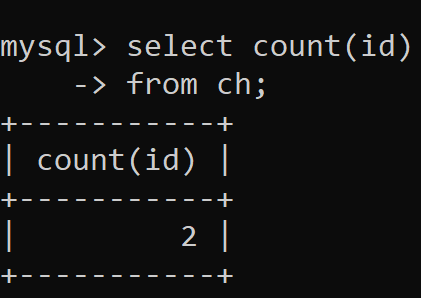
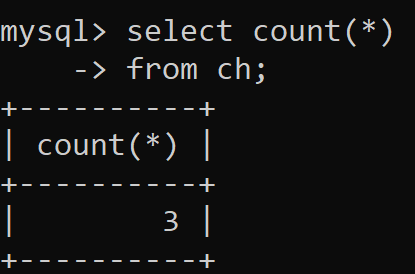
AVG() ustunning o`rtacha qiymatini qaytaradi.

SUM() ustunning umumiy yig`indisini qaytaradi.

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

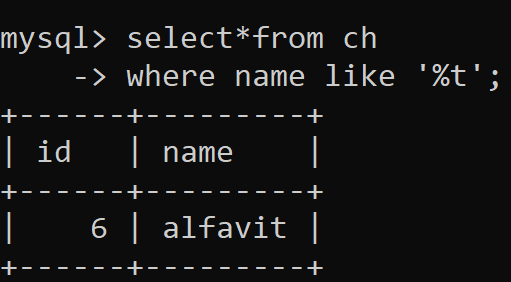
**Eslatma:** NULL qiymatlari e’tiborga olinmaydi.

# LIKE operatori

LIKE o`xshashlikni qidirish uchun ishlatiladi. Belgilari: (%) va (\_)

* Foiz belgisi (%) bitta yoki bir nechta belgilarni ifodalaydi;
* Pastki chiziq (\_) bitta belgini bildiradi;
* Kombinatsiya ikkovi qatnashadi.

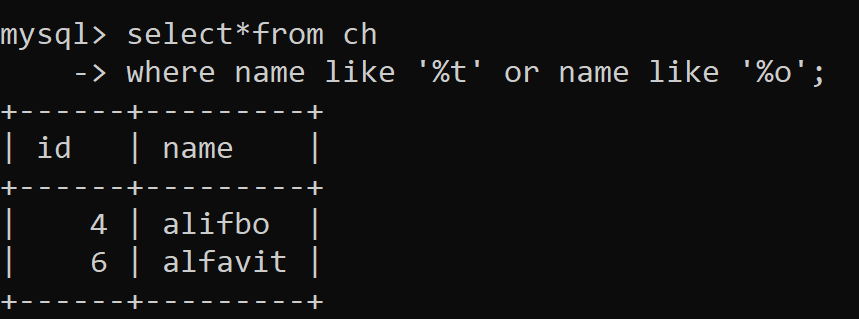
SELECT \* FROM table\_name  
WHERE column LIKE pattern;

SELECT \* FROM Customers  
WHERE CustomerName NOT LIKE ’a%’;

|  |  |
| --- | --- |
| **Like operator** | **Tavsif** |
| LIKE ‘a%’ | "a" harfi bilan boshlanadi |
| LIKE ‘%a’ | "a" bilan tugaydi |
| LIKE ‘%or%’ | So`zda "or" ketma-ketligi mavjud |
| LIKE ‘\_r%’ | Ikkinchi harfi "r" |
| LIKE ‘a\_%’ | “a” bilan boshlangan va uzunligi kamida 2 ta belgidan iborat |
| LIKE ‘a\_\_%’ | “a” bilan boshlanadigan va uzunligi kamida 3 ta belgidan iborat |
| LIKE ‘a%o` | “a” bilan boshlanib, “o” bilan tugaydigan qiymatlarni topadi |
| LIKE ‘%a%o%u’ | “a” bilan boshlanib, o`rtada “o” va “u” bilan tugaydigan qiymatlarni topadi |

|  |  |
| --- | --- |
|  | Foiz belgisini o`z ichiga olgan yozuvlarni topish |
| SELECT \* FROM wildcard WHERE test LIKE ‘**%\%%**’ **ESCAPE ‘\’**; |
|  | Pastki chiziq belgisi bilan boshlangan qiymatlarni topish |
| SELECT x.ksppinm NAME, y.ksppstvl VALUE, x.ksppdesc DESCRIPTION FROM x$ksppi x, x$ksppcv y WHERE x.inst\_id = userenv(‘Instance’) AND y.inst\_id = userenv(‘Instance’) AND x.indx = y.indx AND x.ksppinm like **’\\_b%’ ESCAPE ‘\’** ORDER BY 1; |

**Maslahat:** Siz AND yoki OR operatorlardan foydalangan holda istalgan sonli shartlarni birlashtira olasiz .



# IN operatori

 IN Operator bandda bir nechta qiymatlarni belgilash imkonini beradi. Ya’ni bir nechta OR shartlarning qisqartmasi.

SELECT \* FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

SELECT \* FROM table\_name  
WHERE column\_name IN (*SELECT* STATEMENT);

Misol:

SELECT \* FROM Customers  
WHERE Country IN (‘Germany’, ’France’);

SELECT \* FROM Customers  
WHERE Country NOT IN (‘Paris’,’London’);

Quyida yetkazib beruvchilar bilan mamlakati bir bo`lgan mijozlarni chiqaradi:

SELECT \* FROM Customers  
WHERE Country IN (SELECT Country

FROM Suppliers);

# BETWEEN operatori

BETWEEN ma’lum diapazondagi qiymatlarni chiqaradi.

Qiymatlar: raqamlar, matn yoki sana bo`lishi mumkin.

BETWEEN Operatoriga boshlang`ich va tugatish qiymatlari kiritiladi.

SELECT \* FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

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

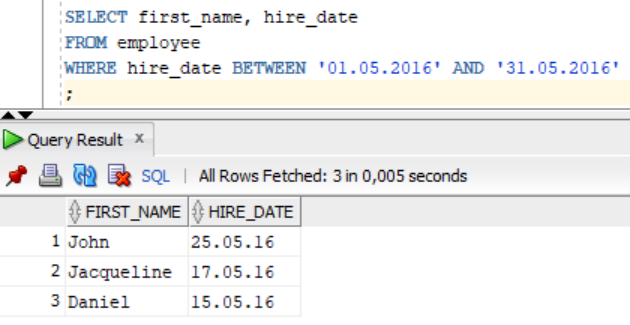
SELECT \* FROM Products  
WHERE Price NOT BETWEEN 10 AND 20;

### BETWEEN va IN misol

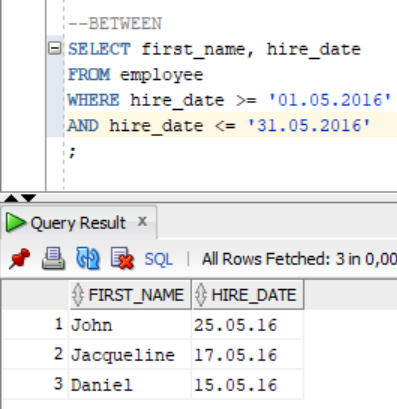
SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20  
AND CategoryID NOT IN (1,2,3);

Quyidagi SQL bayonoti "01-iyul-1996" va "31-iyul-1996" orasida Buyurtma sanasi bilan barcha buyurtmalarni tanlaydi:

SELECT \* FROM employee   
WHERE Hire\_date BETWEEN ’01.05.2016’ AND ’31.05.2016’;



Bunga sinonim:



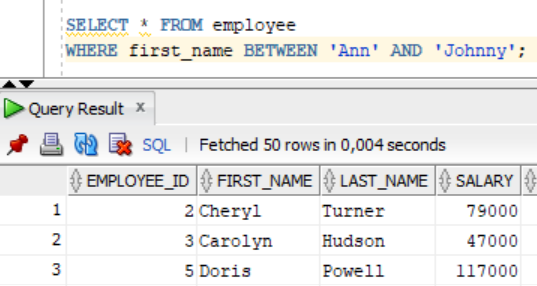
Mashq:

BETWEEN operatordan foydalanib, " Ann " va " Johnny " o`rtasida joylashgan barcha yozuvlarni alifbo tartibida chiqaring.

**SELECT \* FROM** Employee

**WHERE** First\_name **BETWEEN** ‘Ann’ **AND** ‘Johnny’**;**

Bu yerda ProductName ustunida ‘*Ann*’ va ‘*Johnny*’ alfavit bo`yicha tanlab, oraliqdagi so`zlarni ro`yhatda qanday bo`lsa o`z holicha chiqaradi.



# Taxalluslar

* SQL taxalluslari jadvalga yoki jadvaldagi ustunga vaqtinchalik nom berish uchun ishlatiladi.
* Taxalluslar ko`pincha ustun nomlarini o`qishni qulay qilish uchun ishlatiladi.
* Taxallus faqat so`rov davomida mavjuddir.

Taxalluslar quyidagi hollarda foydali bo`lishi mumkin:

* So`rovda bir nechta jadval mavjud bo`lsa
* So`rovda funksiyalar qo`llanilsa
* Ustun nomlari uzun yoki unchalik o`qilmaydigan bo`lsa
* Ikki yoki undan ortiq ustunlar birlashtirilsa

AS kalit so`zi bilan taxallus yaratiladi.

SELECT column\_name AS alias\_name  
FROM table\_name;

Misol:

SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;

**Eslatma:** Agar taxallus nomida boʻshliqlar boʻlsa, qoʻshtirnoqni talab qiladi:

SELECT CustomerName AS Customer,

ContactName AS “Contact Person”  
FROM Customers;

Quyidagi SQL bayonoti 2ta ustunni (shahar va mamlakat) birlashtirgan "Manzil" nomli taxallusni yaratadi:

Qulaylashtirishga misol:

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;

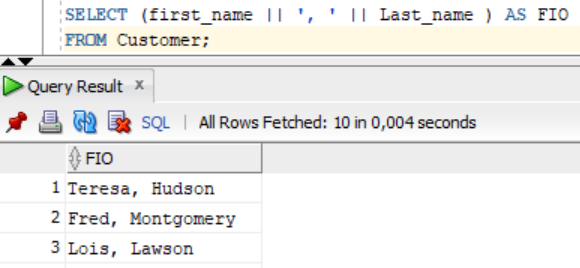
Quyidagi SQL bayonoti yuqoridagi bilan bir xil, ammo taxalluslarsiz:

SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName  
FROM Customers, Orders  
WHERE Customers.CustomerName=‘Around the Horn’ AND Customers.CustomerID=Orders.CustomerID;

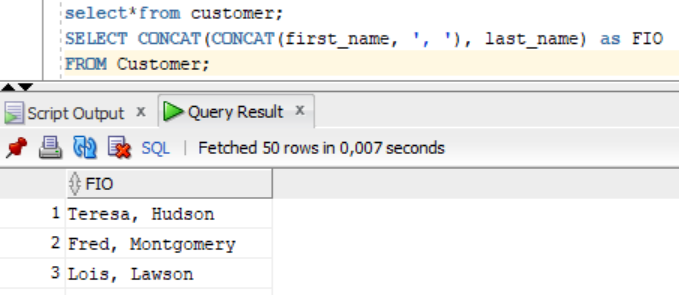
# Concat

Concat. Bir nechta ustunlarning ma’lumotlarini birlashtirib chiqarish:

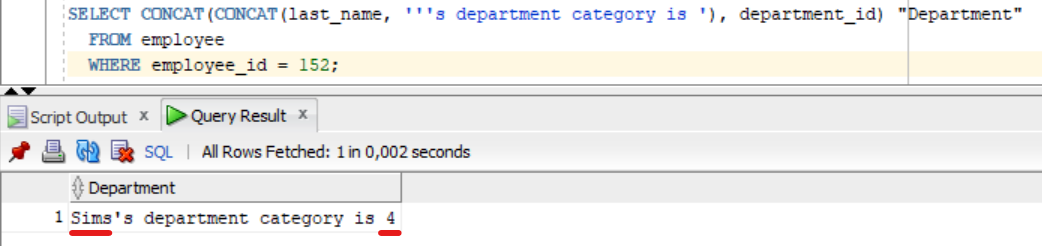
SELECT (first\_name, Last\_name) AS FIO   
FROM Customers;



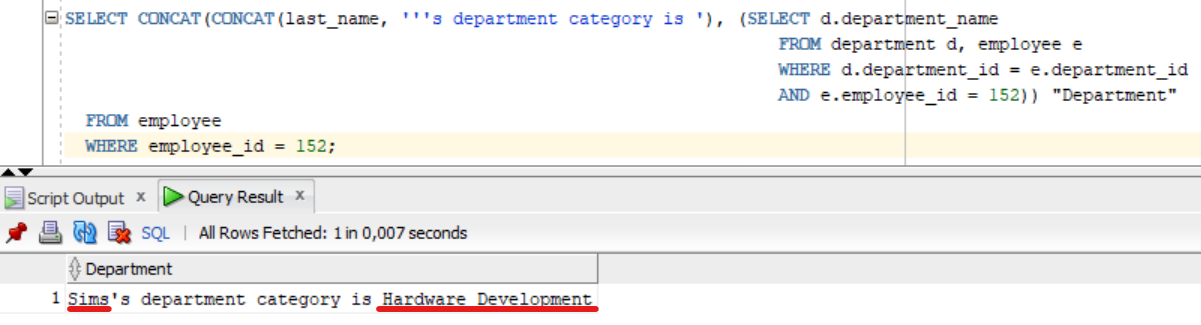
Bunga sinonim:



Boshqa misol:

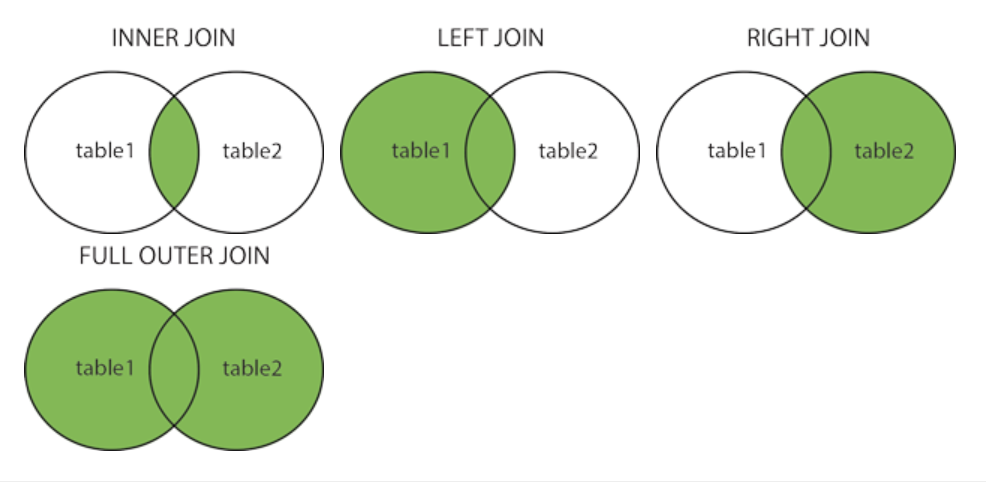
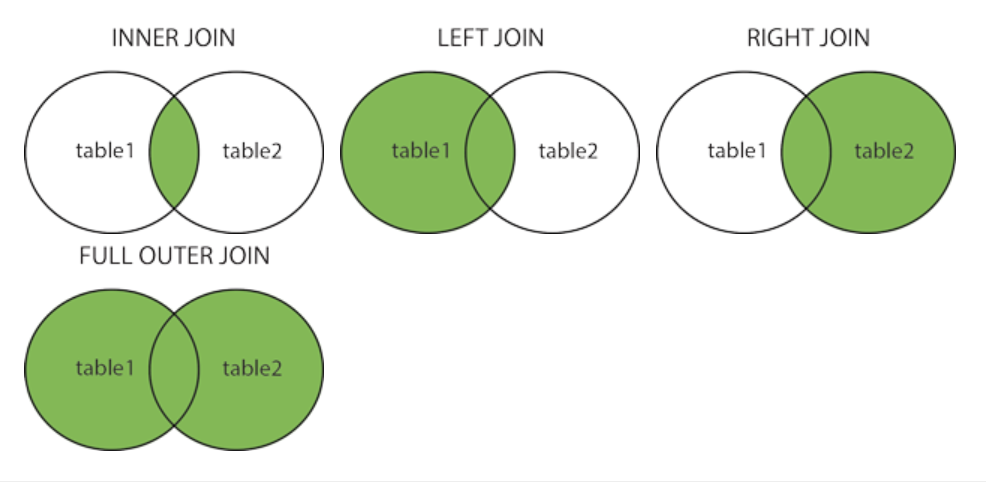


Kengaytirilgan sinonim:



# JOIN

JOIN Ikki yoki undan ortiq jadvallar qatorlarini ular orasidagi tegishli ustun asosida birlashtirish uchun ishlatiladi.



## INNER Join

INNER JOIN 2ta jadvalda mos qiymatlarga ega bo`lgan yozuvlarni chiqaradi.

**SELECT** column\_name(s)  
**FROM** table1  
**INNER** **JOIN** table2**ON** table1.column\_name = table2.column\_name;

### Misol

**SELECT** Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
**FROM** Orders  
**INNER** **JOIN** Customers **ON** Orders.CustomerID=Customers.CustomerID;

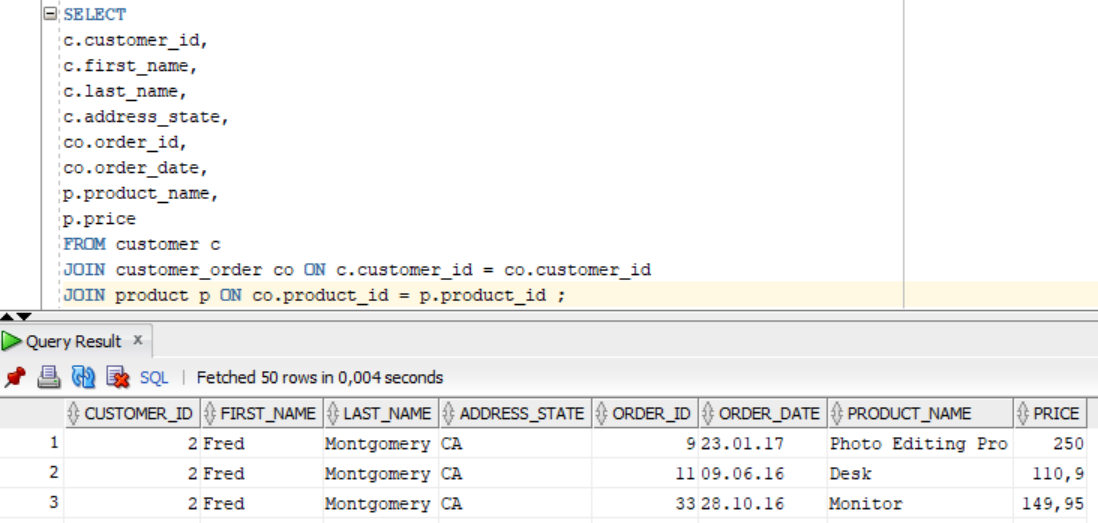
### Misol

Quyidagi so`rov mijoz va jo`natuvchi ma’lumotlari bilan barcha buyurtmalarni tanlaydi:

SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

### Misol

Bu yerda esa Customer Customer\_order va Product jadvallarini bog`liq holda chiqardik:



## SELF Join

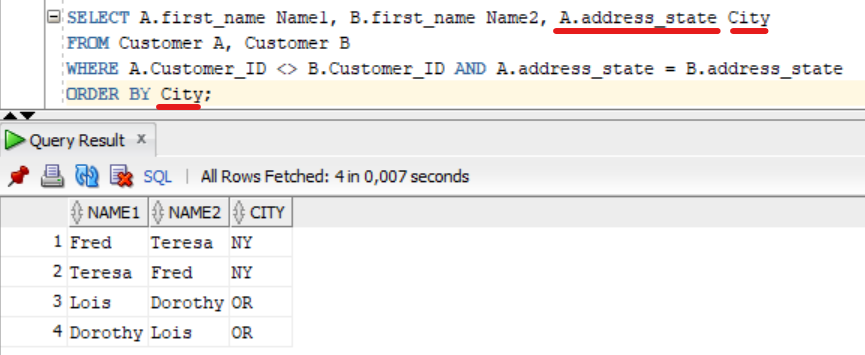
O`z-o`zidan qo`shilish odatiy qo`shilishdir, lekin jadval o`zi bilan birlashtiriladi.

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

*T1* va *T2* bir xil jadval uchun turli xil jadval taxalluslaridir.

Quyidagi so`rov bir shahardan bo`lgan mijozlarga mos keladi:

SELECT A.first\_name Name1, B.first\_name Name2, A.address\_state City   
FROM Customer A, Customer B  
WHERE A.Customer\_ID <> B.Customer\_ID AND A. address\_state = B.address\_state  
ORDER BY City;



# UNION operatori

UNION ikki yoki undan ortiq bayonotlarning natijalar to`plamini birlashtirish uchun ishlatiladi.

* Har bir SELECT bayonotda UNION ustunlar soni bir xil bo`lishi kerak
* Ustunlar ham o`xshash *datatype* ga ega bo`lishi kerak
* Har bir SELECT bayonotdagi ustunlar ham bir xil tartibda bo`lishi kerak

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

UNION faqat turli qiymatlarni chiqaradi, bir xillarni bittasini oladi.

UNION ALL bir xil nusxadagi 2ta qiymatni ham chiqaraveradi.

**Eslatma:** Natijalar to`plamidagi ustun nomlari odatda birinchi SELECT bayonotdagi ustun nomlariga teng bo`ladi.

Quyidagi SQL bayonoti "Mijozlar" va "Yetkazib beruvchilar" jadvalidagi shaharlarni (faqat alohida qiymatlarni) qaytaradi:

### Misol

SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers  
ORDER BY City;

**Eslatma:** Agar ba’zi mijozlar yoki yetkazib beruvchilar bir xil shaharga ega bo`lsa, har bir shahar faqat bir marta ro`yxatga olinadi, chunki UNION faqat turli qiymatlarni tanlaydi.

UNION ALL esa ikki nusxadagi qiymatlarni tanlash uchun ham foydalaniladi !

Quyidagi SQL bayonoti "Mijozlar" va "Yetkazib beruvchilar" jadvalidagi shaharlarni (shuningdek takroriy qiymatlarni) qaytaradi:

### Misol

SELECT City FROM Customers  
UNION ALL  
SELECT City FROM Suppliers  
ORDER BY City;

Quyidagi SQL bayonoti "Mijozlar" va "Yetkazib beruvchilar" jadvalidan Germaniya mamlakati shaharlarini (faqat alohida qiymatlarni) qaytaradi:

### Misol

SELECT City, Country FROM Customers  
WHERE Country=‘Germany’  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country=‘Germany’  
ORDER BY City;

Quyidagi SQL bayonotida barcha mijozlar va yetkazib beruvchilar ro`yxati keltirilgan:

### Misol

SELECT ’Customer’ AS Type, ContactName, City, Country  
FROM Customers  
UNION  
SELECT ’Supplier’, ContactName, City, Country  
FROM Suppliers;

Yuqoridagi "AS Type" ga e’tibor bering - bu taxallus. [SQL taxalluslari](https://www.w3schools.com/sql/sql_alias.asp) jadval yoki ustunga vaqtinchalik nom berish uchun ishlatiladi. Taxallus faqat so`rovning davomiyligi uchun mavjud. Shunday qilib, biz bu yerda "Type" nomli vaqtinchalik ustunni yaratdik, unda aloqa qiluvchi shaxs "Mijoz" yoki "Yetkazib beruvchi" bo`ladi.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **ContactName** | **City** | **Country** |
| Customer | Alejandra Camino | Madrid | Spain |
| Customer | Alexander Feuer | Leipzig | Germany |
| Customer | Ana Trujillo | México D.F. | Mexico |
| Supplier | Anne Heikkonen | Lappeenranta | Finland |
| Supplier | Antonio del Valle | Oviedo | Spain |
| Supplier | Beate Vileid | Sandvika | Norway |

# GROUP BY bayonoti

GROUP BY bir xil qiymatlarga ega boʻlgan qatorlarni sonini aniqlaydi guruhlaydi.

GROUP BY natijalar to`plamini bir yoki bir nechta ustunlar bo`yicha guruhlash uchun ishlatiladi. U ko`pincha agregat funktsiyalar bilan keladi (COUNT(), MAX(), MIN(), SUM(), AVG() )

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

Har bir mamlakatdagi mijozlar sonini + malakat nomini chiqarish

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

Har bir mamlakatdagi mijozlar sonini kattadan kichikka qarab saralash + mamlakat nomi bilan

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

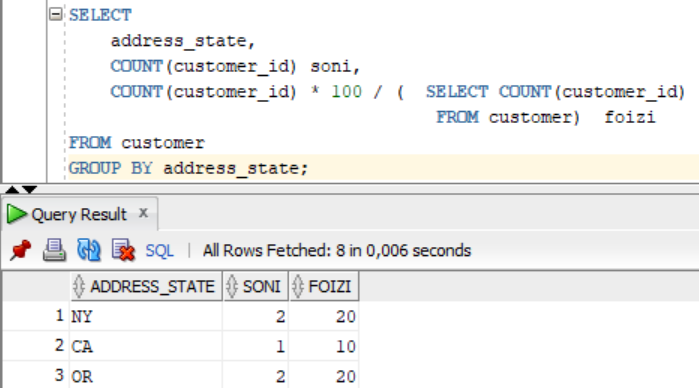
Turli xil jinslar soni va foizini chiqarish:

SELECT GENDER, COUNT(ID),

COUNT(ID)\* 100/ (SELECT COUNT(ID) FROM students) AS foizi

FROM STUDENTS

GROUP BY GENDER;



SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders

FROM Orders  
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
GROUP BY ShipperName;

## HAVING bandi

Ushbu HAVING band SQL-ga qo`shildi, chunki WHERE kalit so`zni yig`ish funktsiyalari bilan ishlatib bo`lmaydi.

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

Quyida har bir mamlakatdagi mijozlar soni ko`rsatilgan. Faqat 5 dan ortiq mijozlari bo`lgan mamlakatlar kiradi:

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

Quyida har bir mamlakatdagi mijozlar soni yuqoridan pastgacha tartiblangan (Faqat 5 dan ortiq mijozlari bo`lgan mamlakatlar kiradi):

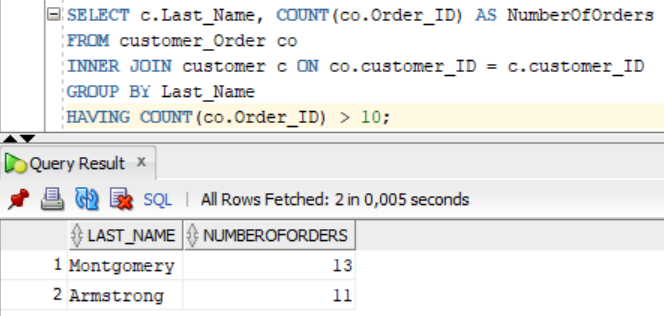
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5  
ORDER BY COUNT(CustomerID) DESC;

Quyida 10 dan ortiq buyurtmalarni ro`yxatdan o`tkazgan xodimlar ro`yxati keltirilgan:

SELECT Employees.LastName,

COUNT(Orders.OrderID) AS NumberOfOrders  
FROM (Orders  
INNER JOIN Employees

ON Orders.EmployeeID = Employees.EmployeeID)  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 10;



Quyida "Davolio" yoki "Fuller" xodimlari 25 dan ortiq buyurtmalarni ro`yxatdan o`tkazgan bo`lsalar ro`yxati keltirilgan:

SELECT Employees.LastName,

COUNT(Orders.OrderID) AS NumberOfOrders  
FROM Orders  
INNER JOIN Employees

ON Orders.EmployeeID = Employees.EmployeeID  
WHERE LastName = ’Davolio` OR LastName = ’Fuller’  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 25;

# EXISTS operatori

EXISTS birorta yozuv mavjudligini tekshirish uchun ishlatiladi.

Agar EXISTS bir yoki bir nechta yozuvlarni qaytarsa, operator TRUE qaytaradi.

Bunda quyi so`rov bajariladi va hech qanday ustun bilan tekshirilmaydi.

### EXISTS sintaksisi

SELECT column\_name(s)  
FROM table\_name  
WHERE EXISTS  
(SELECT column\_name FROM table\_name WHERE condition);

Quyida TRUE qiymatini qaytaradi va mahsulot narxi 20 dan past bo`lgan yetkazib beruvchilarni ro`yxatga oladi:

SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName

FROM Products

WHERE Products.SupplierID = Suppliers.supplierID

AND Price < 20);

Proyektga proyeksiya:

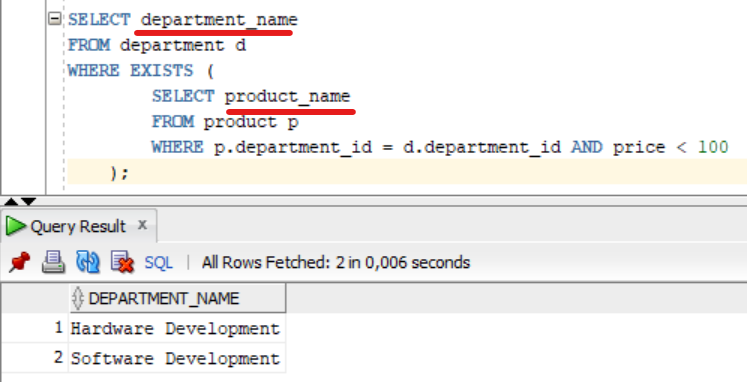
*PRODUCT* jadvalida *department\_id* bilan mos keladigan  kamida bitta yozuv mavjud bo`lgan *department* jadvalidagi *department\_name* larni qaytaradi .

SELECT department\_Name  
FROM department d  
WHERE EXISTS (SELECT Product\_Name

FROM Product p

WHERE p.Department\_ID = d. Department\_ID

AND Price < 100);



# ANY(some) va ALL operatorlari

ANY va ALL operatorlari bitta ustun qiymati va boshqa qiymatlar oralig`ini solishtirish imkonini beradi.

Operator ANY va SOME sinonimdir *(HAR QANDAYI yoki BIRORTASI):*

* natijada mantiqiy qiymatni qaytaradi
* agar quyi soʻrov qiymatlaridan HAR QANDAYI shartga javob bersa, TRUE qiymatini qaytaradi

ANY birorta qiymat uchun amal to`g`ri bo`lsa, shart to`g`ri bo`lishini anglatadi.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ANY  
  (SELECT column\_name  FROM table\_name  WHERE condition);

**Eslatma:** Operator: (=, <>, !=, >, >=, <, <=).

Operator ALL *(BARChA)*:

* natijada mantiqiy qiymatni qaytaradi
* agar quyi so`rovning BARChA qiymatlari shartga javob bersa, TRUE qiymatini qaytaradi
* SELECT, WHERE va HAVING lar bilan ishlatiladi

ALL amal diapazondagi barcha qiymatlar uchun to`g`ri bo`lsagina shart to`g`ri bo`ladi, degan ma’noni anglatadi.

SELECT ALL column\_name(s)  
FROM table\_name  
WHERE condition;

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ALL  
  (SELECT column\_name  FROM table\_name  WHERE condition);

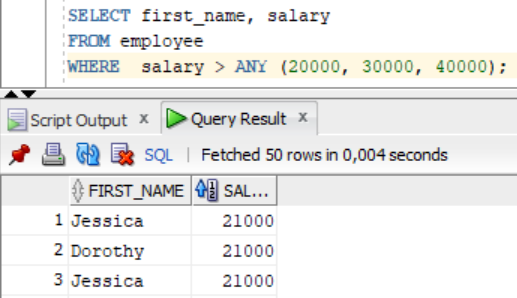
**Eslatma:** Operator: (=, <>, !=, >, >=, <, <=).

Quyida Buyurtma tafsilotlari jadvalidagi HAR QANDAY yozuvlar soni 99 dan katta bo`lsa, Mahsulot nomi ro`yxatini beradi (bu TRUE bo`ladi, chunki Miqdor ustunida 99 dan katta qiymatlar mavjud):

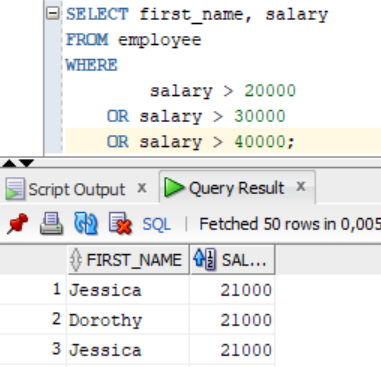
SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity > 99);

Quyidagi SQL iborasi Buyurtma tafsilotlari jadvalida 1000 dan katta miqdorga ega HAR QANDAY yozuvni topsa, Mahsulot nomi ro`yxatini beradi (bu FALSE qaytaradi, chunki Miqdor ustunida 1000 dan katta qiymat yo`q):

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity > 1000);



Sinonim:

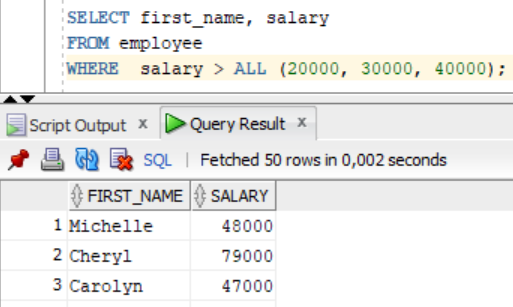


Quyidagi SQL bayonotida BARChA mahsulot nomlari keltirilgan:

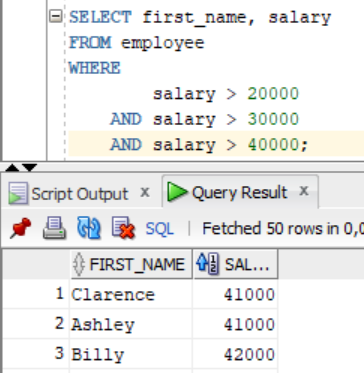
SELECT ALL ProductName  
FROM Products  
WHERE TRUE;

Quyida, Agar Buyurtma Details jadvalidagi Miqdor ustunidagi BARChA yozuvlar 10 ga teng bo`lsa, Mahsulot nomini ko`rsatadi. Bu, albatta, FALSEni qaytaradi, chunki Miqdor ustunida juda ko`p turli qiymatlar mavjud (faqatgina 10 qiymati emas):

SELECT ProductName  
FROM Products  
WHERE ProductID = ALL  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity = 10);



Sinonim:



# INSERT INTO SELECT bayonoti

Bayonot INSERT INTO SELECT bir jadvaldan ma’lumotlarni ko`chiradi va boshqa jadvalga kiritadi.

Bayonot INSERT INTO SELECT manba va maqsadli jadvallardagi ma’lumotlar turlari mos kelishini talab qiladi.

**Eslatma:** Maqsadli jadvaldagi mavjud yozuvlar ta’sir qilmaydi.

Barcha ustunlarni bitta jadvaldan boshqa jadvalga nusxalash:

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

Bitta jadvaldan faqat ba’zi ustunlarni boshqa jadvalga nusxalash:

INSERT INTO table2 (column1, column2, column3, ...)  
SELECT column1, column2, column3, ...  
FROM table1  
WHERE condition;

### Misol

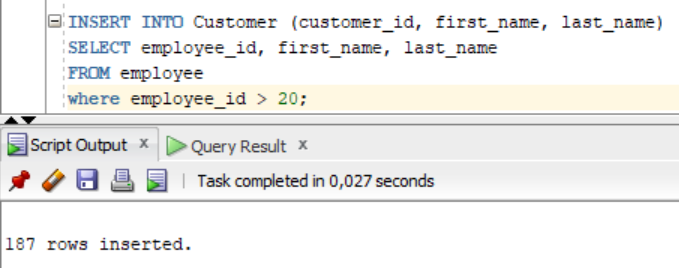
Quyida "xodimlar" ni "Mijozlar" ga ko`chiradi (ma’lumotlar bilan to`ldirilmagan ustunlar NULLni o`z ichiga oladi):

INSERT INTO Customer (customer\_id, first\_name, last\_name)

SELECT employee\_id, first\_name, last\_name

FROM employee

WHERE employee\_id > 20;



Quyida faqat nemis Yetkazib beruvchilarini "Mijozlar" ga ko`chiradi:

INSERT INTO Customers (CustomerName, City, Country)  
SELECT SupplierName, City, Country

FROM Suppliers  
WHERE Country=‘Germany’;

# CASE ifodasi

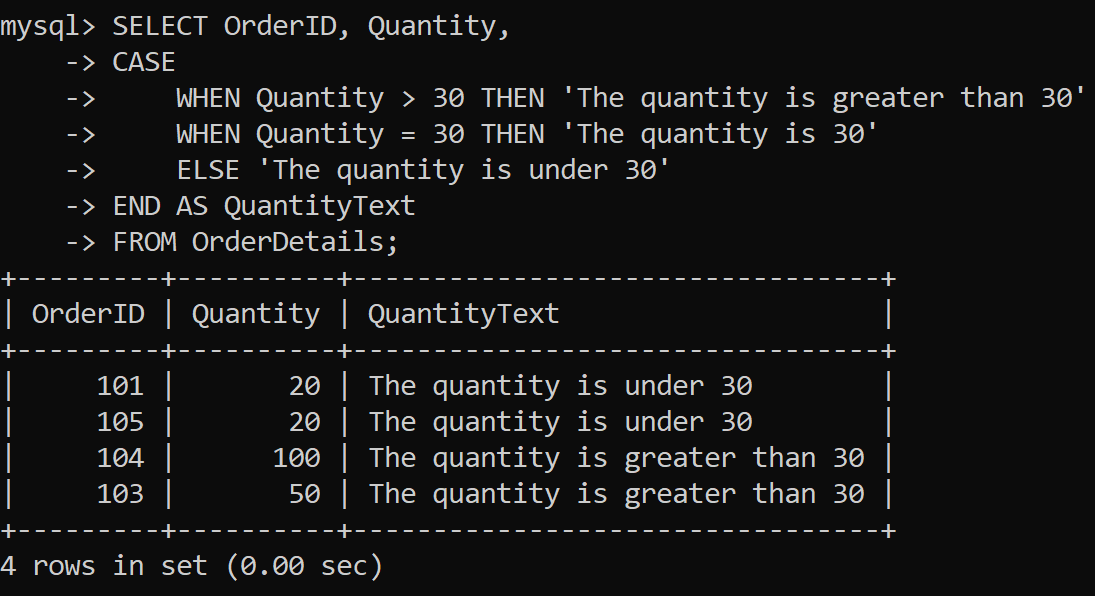
* CASE Ifoda shartlar orqali o`tadi va birinchi shart bajarilganda qiymatni qaytaradi
* Shart to`g`ri bo`lsa, u o`qishni to`xtatadi va natijani qaytaradi.
* Hech qanday shart to`g`ri bo`lmasa, u ELSE bandidagi qiymatni qaytaradi.
* Hech qanday ELSE qism bo`lmasa va hech qanday shartlar to`g`ri bo`lmasa, u *NULL*ni qaytaradi.

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

Quyida shartlardan o`tadi va birinchi shart bajarilganda qiymatni qaytaradi:

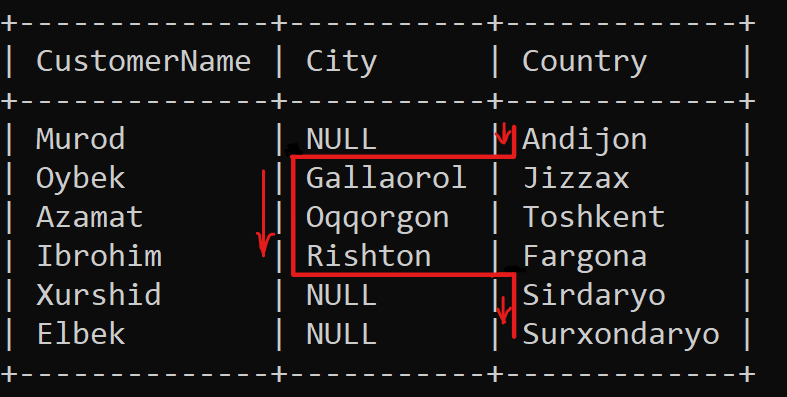
### Misol

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;



Quyida mijozlarga City bo`yicha tartiblab chiqaradi. Agar City NULL bo`lsa, Country bo`yicha tartiblab chiqaradi:

SELECT CustomerName, City, Country  
FROM Customers  
ORDER BY  
 (CASE  
     WHEN City IS NULL THEN Country  
    ELSE City  
 END);



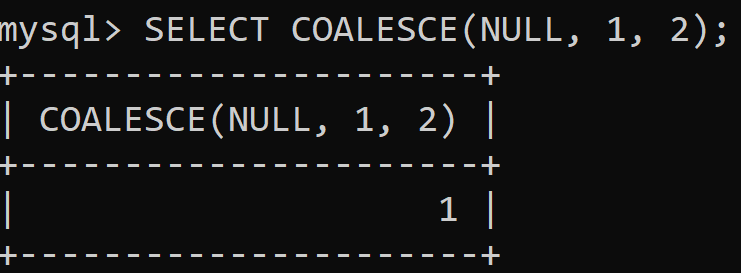
# COALESCE

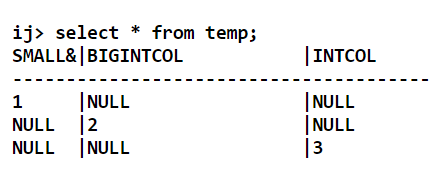
COALESCE() funktsiyasi ro`yxatdagi birinchi NULL bo`lmagan qiymatni qaytaradi.

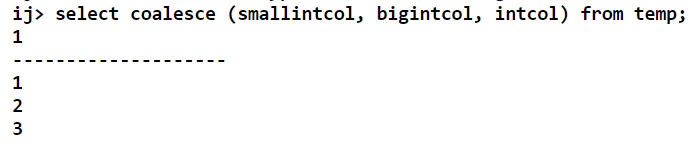
COALESCE(val1, val2, ...., val\_n)

val1, val2, val\_n - Sinov uchun qiymatlar

SELECT COALESCE(NULL, 1, 2);

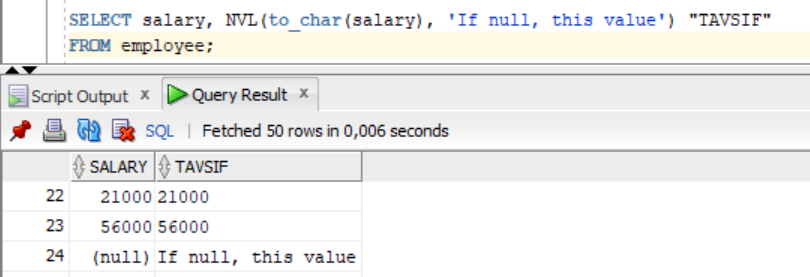






# NVL

NVL => bo`sh qiymatlar o`rnini ixtiyoriy belgilash bilan to`ldirib chiqaradi.



# Comments

Bir qatorli izohlarga 2 ta (--) ishorasi qo`yiladi.

Ko`p qatorli izohlar /\* bilan boshlanadi va \*/ bilan tugaydi .

/\* va \*/ orasidagi har qanday matn e’tiborga olinmaydi.

/\*Select all the columns  
of all the records  
in the Customers table:\*/  
SELECT \* FROM Customers;

Bayonotning faqat bir qismini e’tiborsiz qoldirish uchun /\*\_\_\*/ izohidan ham foydalansa bo`ladi:

SELECT CustomerName, /\*City,\*/ Country FROM Customers;

# OPERATORLAR

### Arifmetik operatorlar

|  |  |
| --- | --- |
| **Operator** | **Ta’rif** |
| + | Qo`shish |
| - | Ayirish |
| \* | Ko`paytirish |
| / | Bo`lish |
| % | Qoldiqli bo`lish |

### Bitwise operatorlari

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A^B** |
| 0 | 0 | 0 |
| 1 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

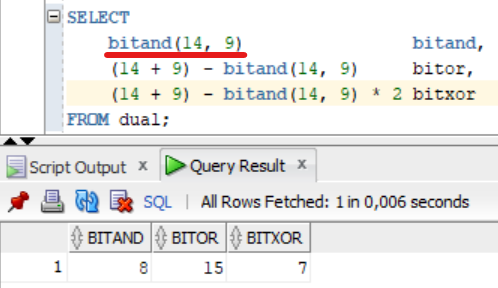
2 lik sanoq sistemasida bajariladigan amallar.

XOR. Qiymati turli xil bo`lgandagina 1 bo`ladi ⮚ XOR ⮚

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Amal** | **Amal** (14,9) | **Sinonim** (a,b) 🡪 | **Yechim** (14,9) |
| & | AND |  | AND(a,b) |  |
| | | OR |  | (a+b) - AND(a,b) | (14+9) – 8 = 15 |
| ^ | XOR |  | (a+b) - AND(a,b)\*2 | (14+9) – 8\*2 = 7 |

### BitAND operatori

Yuqoridagi misollarning bazadagi amaliy ko`rinishi:



### Solishtirish operatorlari

|  |  |
| --- | --- |
| **Operator** | **Ta’rif** |
| = | Teng |
| > | Katta |
| < | Kichik |
| >= | Kichik emas |
| <= | Katta emas |
| <> | Teng emas |
| != | Teng emas |

### Mantiqiy operatorlari

|  |  |
| --- | --- |
| **Operator** | **Ta’rif** |
| ALL | Barcha soʻrov qiymatlari shartga javob bersa, TRUE |
| AND | VA bilan ajratilgan barcha shartlar TRUE bo`lsa, TRUE |
| ANY | Soʻrov qiymatlaridan birortasi shartga javob bersa, TRUE |
| BETWEEN | Operand taqqoslash oralig`ida bo`lsa, TRUE |
| EXISTS | Soʻrov bir yoki bir nechta yozuvni qaytarsa, TRUE |
| IN | Operand ifodalar roʻyxatidan biriga teng boʻlsa, TRUE |
| LIKE | Operand o`xshashlikka mos kelsa, TRUE |
| NOT | Shart(lar) TRUE EMAS bo`lsa, yozuvni ko`rsatadi |
| OR | OR bilan ajratilgan shartlardan biri TRUE bo`lsa, TRUE |
| SOME | Soʻrov qiymatlaridan birortasi shartga javob bersa, TRUE |

# CREATE TABLE. DROP TABLE.

Ma’lumotlar bazasini yaratish: CREATE DATABASE database\_name;

Mavjud bazani o`chirish: DROP DATABASE database\_name;

## JADVAL YARATISH VA O`CHIRISH

***DROP TABLE***

***TRUNCATE TABLE***

***CREATE TABLE***

Yangi jadval yaratish:

CREATE TABLE table\_name (column1 datatype,  
    column2 datatype, ....);

CREATE TABLE Persons ( PersonID int,  
    Name varchar(255) );

***DROP TABLE***

DROP TABLE mavjud jadvalni bazadan o`chiradi.

DROP TABLE table\_name;

***TRUNCATE TABLE***

TRUNCATE TABLE jadval o`chmaydi, ichidagi ma’lumotlar o`chadi

TRUNCATE TABLE table\_name;

TRUNCATE TABLE t\_name ⬄ DELETE FROM t\_name

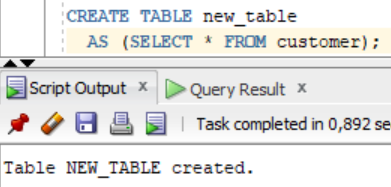
### Mavjud jadval yordamida yangi jadval yaratish

Mavjud jadvalda nusxa olib, yangi jadval yaratish:

[Struktura va ma’lumot olinadi, cheklovlar olinmaydi.]

CREATE TABLE new\_table

AS (SELECT \* FROM old\_table);



# ALTER TABLE bayonoti

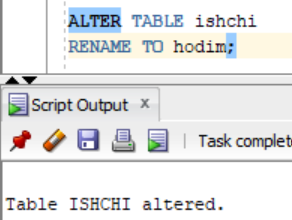
ALTER TABLE jadvalga ustunlarni qo`shish, o`chirish yoki o`zgartirish uchun ishlatiladi.

ALTER TABLE jadvalga cheklovlarni qo`shish va o`chirish uchun ham ishlatiladi.

Jadval nomini o`zgartirish:

**ALTER TABLE** *old\_name*

**RENAME TO** *new\_name*;

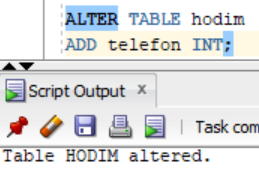


Jadvalga ustun qo`shish:

**ALTER TABLE** table\_name

**ADD** (column\_1 *datatype*,

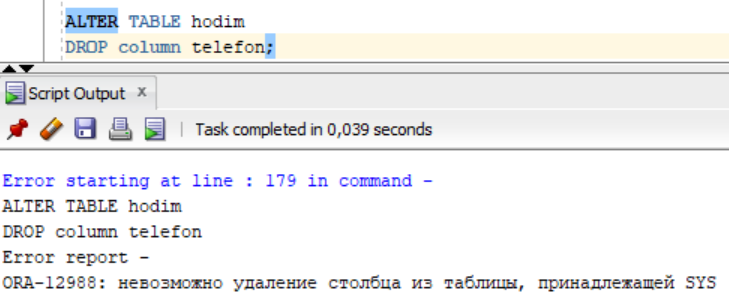
...);



Jadvaldagi ustunni o`chirish:

**ALTER TABLE** table\_name

**DROP COLUMN** *column\_name;*



(Faqatgina SYS bazadan o’chirish mumkin emas,boshqasida m-n)

Ustun nomini o`zgartirish:

**ALTER TABLE** table\_name

**RENAME COLUMN** old*\_name*

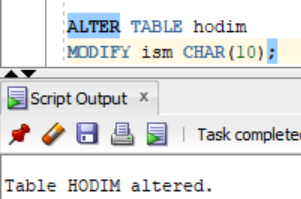
**TO** new*\_name;*

Ustunning ma’lumot tipi va o`lchamini o`zgartirish:

**ALTER TABLE** table\_name

**MODIFY** (column\_1 *datatype(length),*

...);



# CONSTRAINTS (cheklovlar)

Cheklovlar CREATE TABLE bilan jadval tuzilayotganda yoki jadval tuzilgandan keyin ALTER TABLE bilan belgilanishi mumkin.

CREATE TABLE table\_name (  
    column1 datatype *constraint*,  
    column2 datatype *constraint*,  
    ....  
);

* Cheklovlar jadvaldagi ma’lumotlarning ***aniqligi*** va ***ishonchliligi***ni ta’minlaydi.
* Cheklovlar jadvalga kirishi mumkin bo`lgan ma’lumotlar turini cheklash uchun ishlatiladi.
* Agar cheklov va ma’lumotlar harakati o`rtasida buzilish mavjud bo`lsa, harakat bekor qilinadi.
* Cheklovlar ustun darajasi yoki jadval darajasi bo`lishi mumkin. Ustun darajasidagi cheklovlar ustunga, jadval darajasidagi cheklovlar esa butun jadvalga qo`llaniladi.

Odatda SQL da quyidagi cheklovlar qo`llaniladi:

* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ustun NULL qiymatiga ega bo`lmasligini ta’minlaydi
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ustundagi barcha qiymatlar noyob bo`lishini ta’minlaydi
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - NOT NULL va UNIQUE birikmasidir. Jadvaldagi har bir qatorni noyob tarzda aniqlaydi.
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Jadvallar orasidagi aloqalarni buzadigan harakatlarni oldini oladi.
* [ChECK](https://www.w3schools.com/sql/sql_check.asp) - Ustundagi qiymatlar ma’lum bir shartga mosligini ta’minlaydi.
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Agar qiymat belgilanmagan bo`lsa, ustun uchun standart qiymatni o`rnatadi.
* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Ma’lumotlar bazasidan ma’lumotlarni juda tez yaratish va olish uchun foydalaniladi.

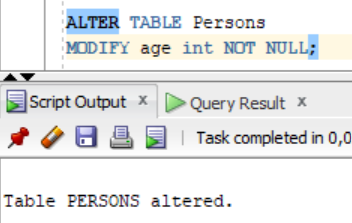
## NOT NULL cheklovi

NOT NULL ustun NULL qiymatlarni qabul qilmaydi. Bu maydonning har doim qiymati mavjud bo`lishini ta’minlaydi, ya’ni maydonga qiymat qo`shmasdan yangi qator qo`sha olmaysiz.

CREATE TABLE Persons (  
    ID int NOT NULL,  
    Name varchar(255) NOT NULL,  
    Age int  
);

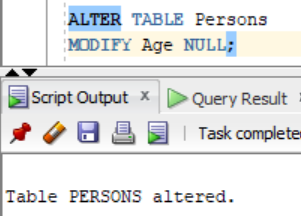
"Persons" jadvalining "Age" ustuniga NOT NULL cheklovini o’rnatish:

ALTER TABLE Persons  
MODIFY Age int NOT NULL;



"Persons" jadvalining "Age" ustunidan NOT NULL cheklovini o’chirish:

ALTER TABLE Persons  
MODIFY Age NULL;



## UNIQUE (UNIKAL) cheklovi

UNIQUE ustundagi barcha qiymatlar noyob bo`lishini ta’minlaydi. Lekin NULL qiymatlarni ham qabul qiladi.

CREATE TABLE Persons (  
    ID int NOT NULL,  
    Age int,  
    UNIQUE (ID)  
);

Cheklovga nom berish va bir nechta ustunlarga UNIQUE cheklovini o`rnatish:

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT UC\_Person UNIQUE (ID, LastName)  
);

UC\_Person <=> UNIQUE\_CONSTRAINT\_Person

UNIQUE mavjud jadvalning "ID" ustuniga cheklov qo’shish:

ALTER TABLE Persons  
ADD UNIQUE (ID);

Cheklovga nom berish va bir nechta ustunlarga UNIQUE cheklovini qo’shish:

ALTER TABLE Persons  
ADD CONSTRAINT UC\_Person UNIQUE (ID,LastName);

UNIQUE cheklovni olib tashlash:

ALTER TABLE Persons  
DROP CONSTRAINT UC\_Person;

UNIQUE cheklovi har bir jadvalda *juda ko`p* bo’lishi mumkin. Lekin PRIMARY KEY cheklovi har bir jadvalda *faqat 1 tadan*  bo`ladi.

## PRIMARY KEY cheklovi

PRIMARY KEY jadvaldagi har bir yozuvni noyobligini ta’minlaydi.

UNIQUE + NOT NULL = PRIMARY KEY

Jadvalda faqat BITTA asosiy kalit bo`lishi mumkin va u bir yoki bir nechta ustunlardan (maydonlardan) iborat bo`lishi mumkin.

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    Age int,  
    PRIMARY KEY (ID)  
);

Cheklovga nom berish va bir nechta ustunlarda PRIMARY KEY cheklovini o`rnatish:

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

Yuqoridagi misolda faqat bitta PRIMARY KEY(PK\_Person) mavjud. Lekin, PRIMARY KEYning qiymati ikki ustundan *(ID + Lastname)* dan iborat.

Mavjud jadvalning "ID" ustuniga PRIMARY KEY cheklovini qo’shish:

ALTER TABLE Persons  
ADD PRIMARY KEY (ID);

Bir nechta ustunlarga PRIMARY KEY cheklovini qo’shish va nom berish:

ALTER TABLE Persons  
ADD CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName);

PRIMARY KEY cheklovini olib tashlash:

ALTER TABLE Persons  
DROP CONSTRAINT PK\_Person;

## FOREIGN KEY cheklovi

FOREIGN KEY cheklovi jadvallar orasidagi aloqalarni buzadigan harakatlarning oldini olish uchun ishlatiladi.

FOREIGN KEY - bu bitta jadvaldagi maydon (yoki maydonlar to`plami), u [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) boshqa jadvalga tegishli.

PRIMARY KEY bo`lgan jadval *Asosiy (Master)* jadval deyiladi.

FOREIGN KEY bo`lgan jadval *Tobe (Detal)* jadvali deyiladi.

FOREIGN KEY cheklovi tashqi kalit ustuniga noto`g`ri ma’lumotlarni kiritishni oldini oladi, chunki u asosiy jadvaldagi qiymatlardan biri bo`lishi kerak.

"Orders" jadvalining "PersonID" ustunida FOREIGN KEY hosil qilish:

CREATE TABLE Orders (  
    OrderID int NOT NULL,

    OrderNumber int NOT NULL,   
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

REFERENCES ⬄ Havolalar ( Link )

Bir nechta ustunlarga FOREIGN KEY cheklovini o`rnatish va nom berish:

CREATE TABLE Orders (  
    OrderID int NOT NULL,

    OrderNumber int NOT NULL,  
    PersonID int,  
    CONSTRAINT PK\_PersonOrder PRIMARY KEY (OrderID),  
    CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID)  
    REFERENCES Persons(PersonID)  
);

"Orders" jadvalining "PersonID" ustuniga FOREIGN KEY cheklovini qo’shish:

ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

Bir nechta ustunlarga FOREIGN KEY cheklovini o’rnatish va nom berish:

ALTER TABLE Orders  
ADD CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

FOREIGN KEY cheklovini olib tashlash:

ALTER TABLE Orders  
DROP CONSTRAINT FK\_PersonOrder;

## ChECK cheklovi (Tekshiruv)

CHECK (tekshiruv) cheklovi ustun uchun qiymat oralig`ini cheklaydi va faqat ma’lum qiymatlargagina ruxsat beradi.

CHECK cheklovi insonning yoshi 18 yoki undan katta bo`lishi kerakligini ta’minlaydi :

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    Age int,  
    ChECK (Age >= 18)  
);

Bir nechta ustunlarda CHECK cheklovini o`rnatish va nom berish:

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255),  
    CONSTRAINT ChK\_Person ChECK (Age>=18 AND City=‘Sandnes’)  
);

Mavjud jadvalning "Age" ustunida ChECK cheklovni yaratish uchun:

ALTER TABLE Persons  
ADD ChECK (Age>=18);

Cheklovga nom berishga ruxsat berish va bir nechta ustunlarda ChECKcheklovini o`rnatish uchun:

ALTER TABLE Persons  
ADD CONSTRAINT ChK\_PersonAge ChECK (Age>=18 AND City=‘Los’);

ChECK cheklovini olib tashlash uchun :

ALTER TABLE Persons  
DROP CONSTRAINT ChK\_PersonAge;

## DEFAULT cheklovi

DEFAULT Cheklov ustun uchun standart qiymatni o`rnatish uchun ishlatiladi.

Agar boshqa qiymat ko`rsatilmagan bo`lsa, standart qiymat barcha yangi yozuvlarga qo`shiladi.

DEFAULT "Persons" jadvalining "City" ustuni uchun "Sandnes" qiymatini o`rnatish:

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255) DEFAULT ’Sandnes’  
);

DEFAULT cheklovi quyidagi kabi funktsiyalardan foydalangan holda tizim qiymatlarini kiritish uchun ham ishlatilishi mumkin: CURRENT\_TIMESTAMP yoki SYSDATE

CREATE TABLE Orders (  
    ID int NOT NULL,   
    OrderDate\_1 Date DEFAULT SYSDATE,

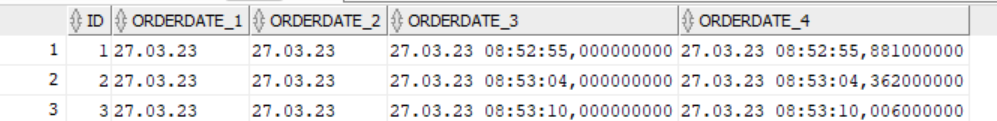
    OrderDate\_2 Date DEFAULT CURRENT\_TIMESTAMP,

    OrderDate\_3 Timestamp DEFAULT SYSDATE,

    OrderDate\_4 Timestamp DEFAULT CURRENT\_TIMESTAMP,  
);

Timestamp ⬄ sana va vaqtni kiritadi:"dd.mm.yy hh.mm.ss"

CURRENT\_TIMESTAMP ⬄ SYSDATE ⬄ Joriy sanani avtomatik kiritadi.



Mavjud jadvalning "City" ustunida DEFAULT cheklovini qo’shish:

ALTER TABLE Persons  
MODIFY City DEFAULT 'Sandnes';

DEFAULT cheklovini o’chirish:

ALTER TABLE Persons  
MODIFY City DEFAULT NULL;

# INDEX bayonoti

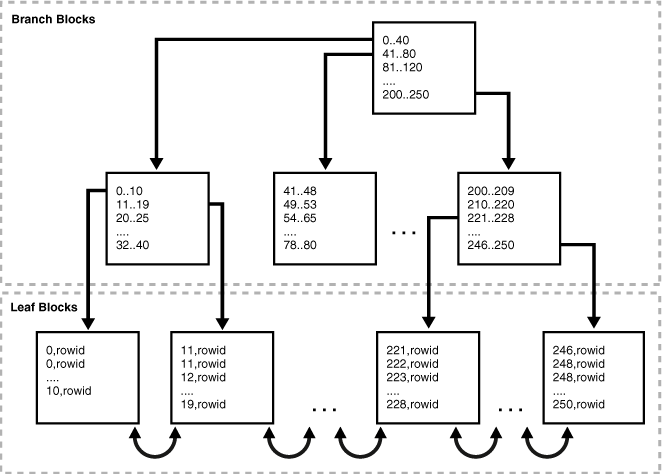
Indekslar ma’lumotlar bazasidan ma’lumotlarni tezroq, samarador olish uchun ishlatiladi. Foydalanuvchilar indekslarni ko`ra olmaydi, ular faqat qidiruv/so`rovlarni tezlashtirish uchun ishlatiladi.

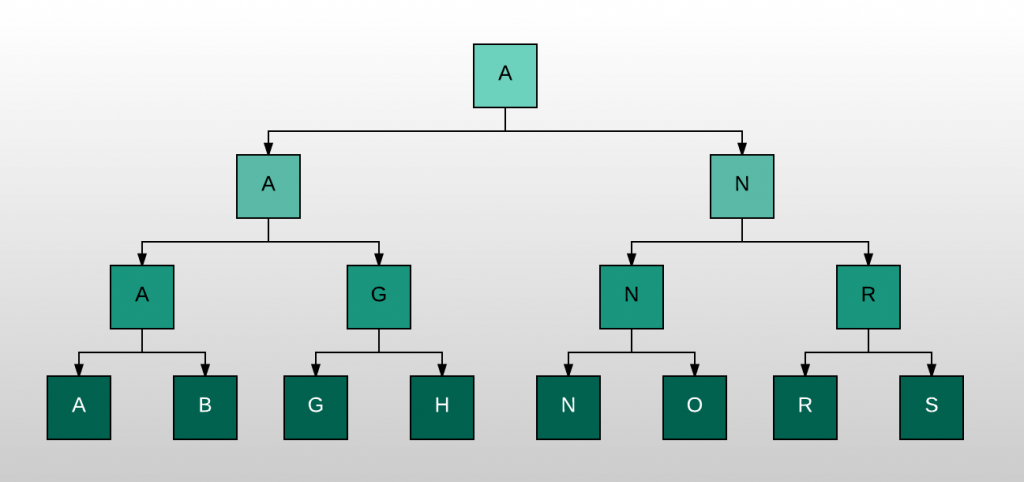
**Eslatma:** Faqatgina tez-tez qidiriladigan ustunlar bo`yicha indekslarni yarating.

Indeks turlari:

* Normal indexes. (Oracle Database creates B-tree indexes.)
* **Bitmap indexes**,  bitmap sifatida kalit qiymat bilan bog'langan qatorlarni saqlaydi
* **Partitioned indexes**, Jadvalning indekslangan ustun(lar)ida ko'rinadigan har bir qiymat uchun yozuvni o'z ichiga olgan bo'limlardan iborat bo'lingan **indekslar**
* **Function-based indexes** ular ifodalarga asoslangan. Ular sizga ifoda tomonidan qaytarilgan qiymatni baholovchi so'rovlarni yaratishga imkon beradi, bu esa o'z navbatida o'rnatilgan yoki foydalanuvchi tomonidan belgilangan funktsiyalarni o'z ichiga olishi mumkin.
* **Domain indexes**, bu turdagi ilovaga xos indeks namunalari *indextype*

### Normal indeks yaratish:





CREATE [UNIQUE] INDEX index\_name

ON table\_name (column1, column2, ... column\_n);

Misol:

CREATE INDEX supplier\_idx

ON supplier (supplier\_name, city);

### CREATE INDEX sintaksisi

Jadvalda indeks yaratish:

[Ikki nusxadagi qiymatlarga ruxsat beriladi]

CREATE INDEX index\_name  
ON table\_name (column1, column2, ...);

Jadvalda noyob indeks yaratish:

CREATE UNIQUE INDEX index\_name  
ON table\_name (column1, column2, ...);

"Persons" jadvalidagi "Last\_name" ustunida "idx\_lastname" nomli indeks yaratish:

CREATE INDEX idx\_lastname  
ON Persons (LastName);

Bir nechta ustunlar uchun bitta indeks yaratish:

CREATE INDEX idx\_pname  
ON Persons (LastName, FirstName);

DROP INDEX jadvaldagi indeksni o`chirish uchun ishlatiladi.

DROP INDEX index\_name;

### Funksiyaga asoslangan indeks yaratish:

CREATE [UNIQUE] INDEX index\_name

ON table\_name (function1, function2, ... function\_n);

Misol:

CREATE INDEX supplier\_idx

ON supplier (UPPER(supplier\_name));

Funksiyali indeks yaratishdan oldin ushbu ustunda NULL qiymatlar yo’qligini tekshirib, ishonch hosil qiling.

### Indeks nomini o'zgartirish

ALTER INDEX index\_name

RENAME TO new\_index\_name;

Misol:

ALTER INDEX supplier\_idx

RENAME TO supplier\_index\_name;

### Indeksni o’chirish

DROP INDEX index\_name;

Misol:

DROP INDEX supplier\_idx;

# AUTO INCREMENT ⬄ SEQUENCE

Sequence - bu Oracle'da raqamlar ketma-ketligini yaratish uchun ishlatiladigan ob'ekt. Bu asosiy kalit sifatida ishlash uchun noyob raqam yaratishda kerak bo’ladi.

CREATE SEQUENCE sequence\_name

MINVALUE value

MAXVALUE value

START WITH value

INCREMENT BY value

CACHE value; / NOCACHE;

Misol:

CREATE SEQUENCE supplier\_seq

MINVALUE 1

MAXVALUE 999999999999999999999999999

START WITH 1

INCREMENT BY 1

NOCACHE;

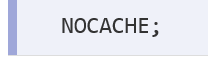
Bu supplier\_seq deb nomlangan ketma-ketlik ob'ektini yaratadi. U ishlatadigan birinchi tartib raqami 1 va har bir keyingi raqam 1 ga oshadi. U ishlash uchun 20 tagacha qiymatni keshlaydi.

### CACHE va NOCACHE farqi

 Ketma-ketlikka nisbatan, *CACHE* (kesh) opsiyasi tezroq kirish uchun xotirada qancha ketma-ketlik qiymatlari saqlanishini belgilaydi.



*NOCACHE* ketma-ketlik qiymatlarining hech biri xotirada saqlanmasligini anglatadi.



### Nextval

Ushbu ketma-ketlik ob'ektidan qiymatni qanday olish mumkinligini ko'rib chiqamiz. Bizga nextval kerak .

supplier\_seq.NEXTVAL;

Bu supplier\_seq dan keyingi qiymatni oladi.

Misol:

INSERT INTO suppliers (supplier\_id, supplier\_name)

VALUES (supplier\_seq.NEXTVAL, 'Kraft Foods');

### Sequence ni o’chirish:

DROP SEQUENCE sequence\_name;

Misol:

DROP SEQUENCE supplier\_seq;

### LASTVALUE ni o’zgartirish

Sequence ni oxirgi qiymati 100 bo'lsa, keyingi qiymatni 225 o’zgartirish uchun:

ALTER SEQUENCE seq\_name

INCREMENT BY 124;

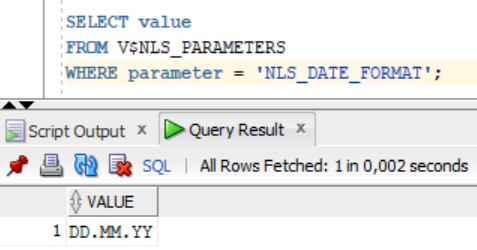
SELECT seq\_name.nextval FROM dual;

ALTER SEQUENCE seq\_name

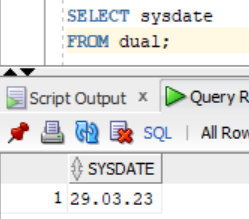
INCREMENT BY 1;

# SANA FORMATINI KO’RISH VA O’ZGARTIRISH

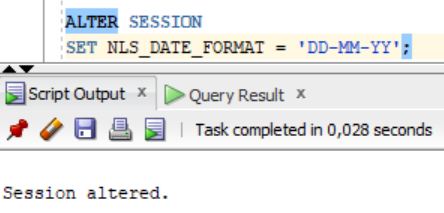
Sana formatini ko’rish:



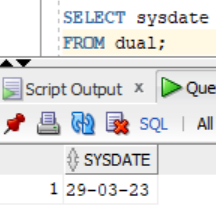
Hozirgi sanani ko’rish:



Sana formatini o’zgartirish:



Hozirgi sanani tekshiramiz:



# DATES

Sanalar bilan ishlashda eng qiyin narsa bu siz kiritmoqchi bo`lgan sana formati bazadagi sana ustunining formatiga mos kelishiga ishonch hosil qilishdir.

Maʼlumotlaringizda faqat sana qismi boʻlsa, soʻrovlaringiz kutilganidek ishlaydi. Biroq, agar vaqt qismi ishtirok etsa, u yanada murakkablashadi.

## sana ma’lumot turlari

Sana yoki sana/vaqt qiymatini saqlash uchun ma’lumot turlari:

* DATE- YYYY-MM-DD formati
* DATETIME- format: YYYY-AA-KK HH:MI:SS
* TIMESTAMP- format: YYYY-AA-KK HH:MI:SS
* YEAR- YYYY yoki YY formati

|  |  |  |
| --- | --- | --- |
| **OrderId** | **ProductName** | **OrderDate** |
| 1 | Geitost | 2008-11-11 |
| 2 | Camembert Pierrot | 2008-11-09 |
| 3 | Mozzarella di Giovanni | 2008-11-11 |

Yuqoridagi jadvaldan "2008-11-11" sanali yozuvlarni chiqarishimiz uchun:

SELECT \* FROM Orders

WHERE OrderDate=‘2008-11-11’

|  |  |  |
| --- | --- | --- |
| **OrderId** | **ProductName** | **OrderDate** |
| 1 | Geitost | 2008-11-11 |
| 3 | Mozzarella di Giovanni | 2008-11-11 |

**Eslatma:** Agar vaqt komponenti bo`lmasa, ikkita sanani osongina solishtirish mumkin!

|  |  |  |
| --- | --- | --- |
| **OrderId** | **ProductName** | **OrderDate** |
| 1 | Geitost | 2008-11-11 13:23:44 |
| 2 | Camembert Pierrot | 2008-11-09 15:45:21 |
| 3 | Mozzarella di Giovanni | 2008-11-11 11:12:01 |

Agar biz SELECT yuqoridagi kabi bir xil bayonotdan foydalansak:

SELECT \* FROM Orders

WHERE OrderDate=‘2008-11-11’

hech qanday natijaga erisha olmaymiz! Buning sababi, so`rov faqat vaqt qismi bo`lmagan sanalarni qidiradi.

**Maslahat:** So`rovlaringizni sodda va oson saqlash uchun, agar kerak bo`lmasa, sanalarda vaqt komponentlaridan foydalanmang!

# VIEWS

View – bu natijalar to`plamiga asoslangan virtual jadval.

VIEW da xuddi haqiqiy jadval kabi qatorlar va ustunlar mavjud.  VIEW dagi maydonlar ma’lumotlar bazasidagi bir yoki bir nechta haqiqiy jadvallarning maydonlaridir.

VIEW ga SQL iboralari va funksiyalarini qo`shishingiz va ma’lumotlarni xuddi bitta jadvaldan olingandek taqdim etishingiz mumkin.

CREATE VIEW *view\_name* AS  
SELECT *column1*, *column2*, ...  
FROM *table\_name*  
WHERE *condition*;

**Eslatma:** VIEW har doim eng so`nggi ma’lumotlarni ko`rsatadi! Ma’lumotlar bazasi mexanizmi har safar foydalanuvchi so`raganida VIEW ni qayta yaratadi.

CREATE VIEW Brazil\_Customers AS  
SELECT CustomerName, ContactName  
FROM Customers  
WHERE Country = ’Brazil’;

Yuqoridagi VIEW ni quyidagicha chiqarishimiz mumkin:

SELECT \* FROM Brazil\_Customers;

"Mahsulotlar" jadvalidagi o`rtacha narxdan yuqori narxga ega har bir mahsulotni tanlaydigan VIEW ni yaratish uchun:

CREATE VIEW Products\_Above\_Average\_Price AS  
SELECT ProductName, Price  
FROM Products  
WHERE Price > (SELECT AVG(Price) FROM Products);

Yuqoridagi VIEW ni quyidagicha chiqarishimiz mumkin:

SELECT \* FROM Products\_Above\_Average\_Price;

## VIEW ni yangilash

VIEW CREATE OR REPLACE VIEW bilan yangilanishi mumkin.

CREATE OR REPLACE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

Mavjud bo`lgan "Brazil\_Customers" view iga "City" ustunini qo`shish uchun:

CREATE OR REPLACE VIEW Brazil\_Customers AS  
SELECT CustomerName, ContactName, City  
FROM Customers  
WHERE Country = ’Brazil’;

View DROP VIEW bilan o`chiriladi.

DROP VIEW view\_name;