ORACLE®

Owner:

Azamat Nishonov

SQL Developer

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

Content

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:

```
INSERT ALL
INTO table_name1(col1, col2) VALUES (val1, val2)
INTO table_name2(col1, col2) VALUES (val1, val2);
```

Misol:

```
INSERT ALL
INTO employee(employee_id, first_name, last_name, department_id) VALUES (303, 'Mark', 'Spencer', 4)
INTO employee(employee_id, first_name, last_name, department_id) VALUES (304, 'Simone', 'Fletcher', 3)
INTO employee(employee_id, first_name, last_name, department_id) VALUES (305, 'Alison', 'Smith', 8)
SELECT * FROM dual;
```

Script Output × Query Result ×

Task completed in 0,04 seconds

```
3 rows inserted.
```

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_name
ORDER 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.

```
mysql> select*from ch;
+------+
| id   |
+------+
|    4 |
|    6 |
| NULL |
+------+
```

```
mysql> select avg(id)
    -> from ch;
+----------+
| avg(id)  |
+----------+
|   5.0000 |
+----------+
```

```
mysql> select count(id)
    -> from ch;
+-----------+
| count(id) |
+-----------+
|         2 |
+-----------+
```

```
mysql> select count(*)
    -> from ch;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
```

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.

```
mysql> select*from ch
    -> where name like '%t';
+------+---------+
| id   | name    |
+------+---------+
|    6 | alfavit |
+------+---------+
```

```
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
`SELECT * FROM wildcard WHERE test LIKE '%\%%' ESCAPE '\';`	Foiz belgisini o`z ichiga olgan yozuvlarni 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;`	Pastki chiziq belgisi bilan boshlangan qiymatlarni topish

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

```
mysql> select*from ch
    -> where name like '%t' or name like '%o';
+------+---------+
| id   | name    |
+------+---------+
|    4 | alifbo  |
|    6 | alfavit |
+------+---------+
```

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’;
```

```
SELECT first_name, hire_date
FROM employee
WHERE hire_date BETWEEN '01.05.2016' AND '31.05.2016'
;
```

Query Result

All Rows Fetched: 3 in 0,005 seconds

	FIRST_NAME	HIRE_DATE
1	John	25.05.16
2	Jacqueline	17.05.16
3	Daniel	15.05.16

Bunga sinonim:

```
SELECT first_name, hire_date
FROM employee
WHERE hire_date >= '01.05.2016'
AND hire_date <= '31.05.2016'
;
```

Query Result

All Rows Fetched: 3 in 0,00

	FIRST_NAME	HIRE_DATE
1	John	25.05.16
2	Jacqueline	17.05.16
3	Daniel	15.05.16

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.

```
SELECT * FROM employee
WHERE first_name BETWEEN 'Ann' AND 'Johnny';
```

Query Result

SQL | Fetched 50 rows in 0,004 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
1	2	Cheryl	Turner	79000
2	3	Carolyn	Hudson	47000
3	5	Doris	Powell	117000

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 bo‘shliqlar bo‘lsa, qo‘shtirnoqni 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;
```

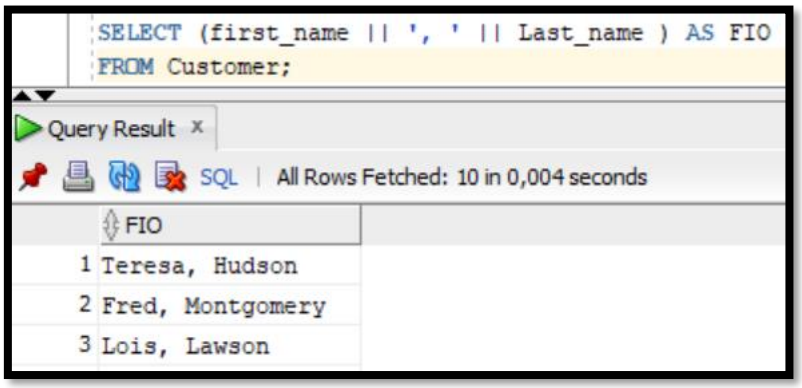
Quyidagi SQL bayonoti yuqoridagi bilan bir xil, ammo taxalluslarsiz:

```
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:

```
select*from customer;
SELECT CONCAT(CONCAT(first_name, ', '), last_name) as FIO
FROM Customer;
```

Script Output | Query Result

SQL | Fetched 50 rows in 0,007 seconds

	FIO
1	Teresa, Hudson
2	Fred, Montgomery
3	Lois, Lawson

Boshqa misol:

```
SELECT CONCAT(CONCAT(last_name, '''s department category is '), department_id) "Department"
  FROM employee
  WHERE employee_id = 152;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0,002 seconds

	Department
1	Sims's department category is 4

Kengaytirilgan sinonim:

```
SELECT CONCAT(CONCAT(last_name, '''s department category is '), (SELECT d.department_name
                                                                  FROM department d, employee e
                                                                  WHERE d.department_id = e.department_id
                                                                  AND e.employee_id = 152)) "Department"
  FROM employee
  WHERE employee_id = 152;
```

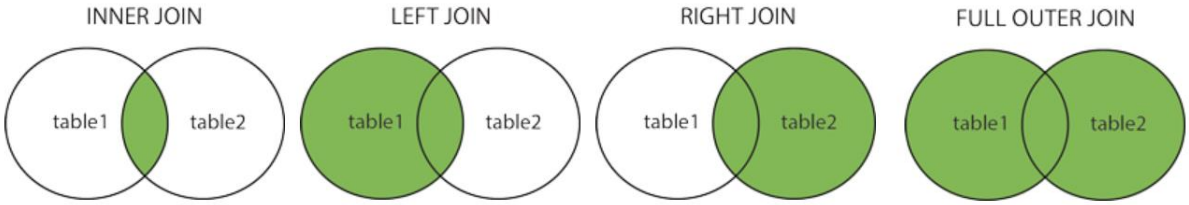
Script Output | Query Result

SQL | All Rows Fetched: 1 in 0,007 seconds

	Department
1	Sims's department category is Hardware Development

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:

```
SELECT
c.customer_id,
c.first_name,
c.last_name,
c.address_state,
co.order_id,
co.order_date,
p.product_name,
p.price
FROM customer c
JOIN customer_order co ON c.customer_id = co.customer_id
JOIN product p ON co.product_id = p.product_id ;
```

Query Result

SQL | Fetched 50 rows in 0,004 seconds

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS_STATE	ORDER_ID	ORDER_DATE	PRODUCT_NAME	PRICE
1	2	Fred	Montgomery	CA	9	23.01.17	Photo Editing Pro	250
2	2	Fred	Montgomery	CA	11	09.06.16	Desk	110,9
3	2	Fred	Montgomery	CA	33	28.10.16	Monitor	149,95

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;
```

```
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;
```

Query Result

SQL | All Rows Fetched: 4 in 0,007 seconds

	NAME1	NAME2	CITY
1	Fred	Teresa	NY
2	Teresa	Fred	NY
3	Lois	Dorothy	OR
4	Dorothy	Lois	OR

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 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 bo‘lgan 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
    address_state,
    COUNT(customer_id) soni,
    COUNT(customer_id) * 100 / (  SELECT COUNT(customer_id)
                                   FROM customer)  foizi
FROM customer
GROUP BY address_state;
```

Query Result

SQL | All Rows Fetched: 8 in 0,006 seconds

	ADDRESS_STATE	SONI	FOIZI
1	NY	2	20
2	CA	1	10
3	OR	2	20

```
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 condition
ORDER 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
```

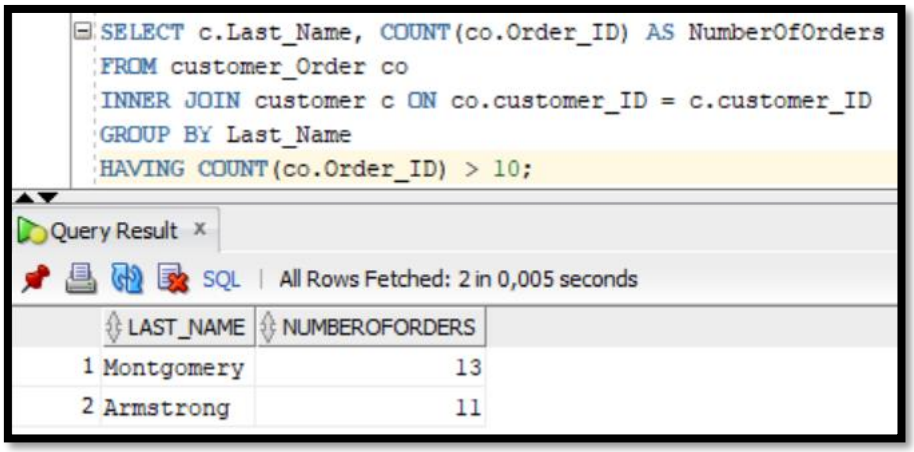
```
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

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);
```

```
SELECT department_name
FROM department d
WHERE EXISTS (
        SELECT product_name
        FROM product p
        WHERE p.department_id = d.department_id AND price < 100
    );
```

Query Result

SQL | All Rows Fetched: 2 in 0,006 seconds

	DEPARTMENT_NAME
1	Hardware Development
2	Software Development

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 so'rov 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);
```

```
SELECT first_name, salary
FROM employee
WHERE  salary > ANY (20000, 30000, 40000);
```

Script Output | Query Result

SQL | Fetched 50 rows in 0,004 seconds

	FIRST_NAME	SAL...
1	Jessica	21000
2	Dorothy	21000
3	Jessica	21000

Sinonim:

```
SELECT first_name, salary
FROM employee
WHERE
        salary > 20000
    OR salary > 30000
    OR salary > 40000;
```

Script Output | Query Result

SQL | Fetched 50 rows in 0,005

	FIRST_NAME	SAL...
1	Jessica	21000
2	Dorothy	21000
3	Jessica	21000

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);
```

```
SELECT first_name, salary
FROM employee
WHERE  salary > ALL (20000, 30000, 40000);
```

Script Output | Query Result

SQL | Fetched 50 rows in 0,002 seconds

	FIRST_NAME	SALARY
1	Michelle	48000
2	Cheryl	79000
3	Carolyn	47000

Sinonim:

```
SELECT first_name, salary
FROM employee
WHERE
        salary > 20000
    AND salary > 30000
    AND salary > 40000;
```

Script Output | Query Result

SQL | Fetched 50 rows in 0,0

	FIRST_NAME	SAL...
1	Clarence	41000
2	Ashley	41000
3	Billy	42000

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 table1
WHERE condition;
```

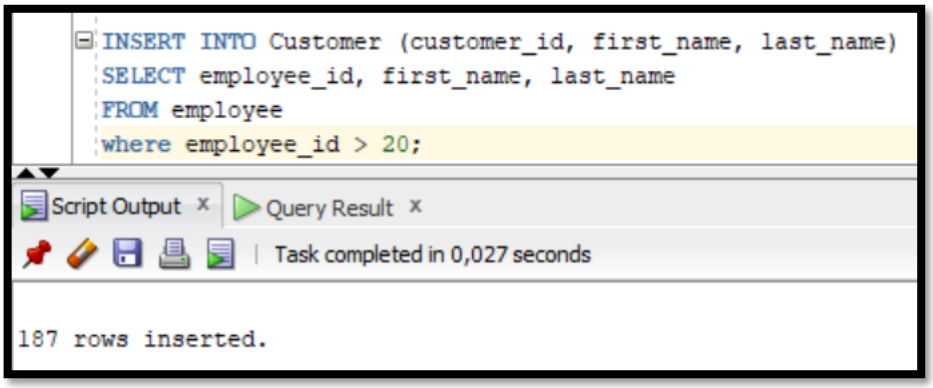
Bitta jadvaldan faqat ba’zi ustunlarni boshqa jadvalga nusxalash:

```
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;
```

```
mysql> 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;
+---------+----------+--------------------------------+
| OrderID | Quantity | QuantityText                   |
+---------+----------+--------------------------------+
|     101 |       20 | The quantity is under 30       |
|     105 |       20 | The quantity is under 30       |
|     104 |      100 | The quantity is greater than 30 |
|     103 |       50 | The quantity is greater than 30 |
+---------+----------+--------------------------------+
4 rows in set (0.00 sec)
```

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);
```

CustomerName	City	Country
Murod	NULL	Andijon
Oybek	Gallaorol	Jizzax
Azamat	Oqqorgon	Toshkent
Ibrohim	Rishton	Fargona
Xurshid	NULL	Sirdaryo
Elbek	NULL	Surxondaryo

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);
```

```
mysql> SELECT COALESCE(NULL, 1, 2);
+----------------------+
| COALESCE(NULL, 1, 2) |
+----------------------+
|                    1 |
+----------------------+
```

```
ij> select * from temp;
SMALL&|BIGINTCOL           |INTCOL
---------------------------------------
1     |NULL                |NULL
NULL  |2                   |NULL
NULL  |NULL                |3
```

```
ij> select coalesce (smallintcol, bigintcol, intcol) from temp;
1
--------------------
1
2
3
```

NVL

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

```
SELECT salary, NVL(to_char(salary), 'If null, this value') "TAVSIF"
FROM employee;
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0,006 seconds

	SALARY	TAVSIF
22	21000	21000
23	56000	56000
24	(null)	If null, this value

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

2 lik sanoq sistemasida bajariladigan amallar.

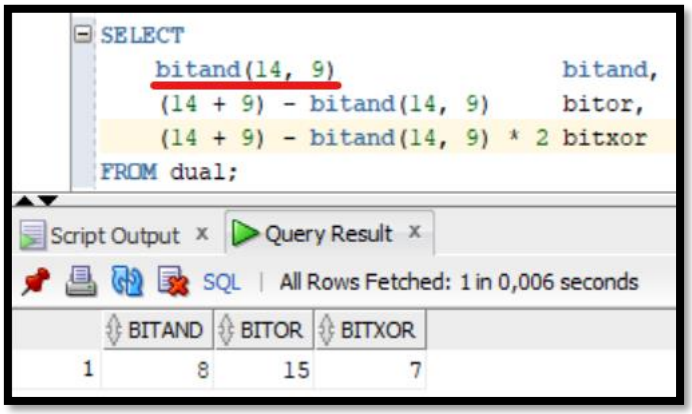
XOR. Qiymati turli xil bo`lgandagina 1 bo`ladi ➢ XOR ➢

Operator	Amal	Amal (14,9)	Sinonim (a,b) →	Yechim (14,9)
&	AND	1110 1001 1000 = 8	AND(a,b)	1110 1001 1000 = 8
|	OR	1110 1001 1111 = 15	(a+b) - AND(a,b)	(14+9) – 8 = 15
^	XOR	1110 1001 0111 = 7	(a+b) - AND(a,b)*2	(14+9) – 8*2 = 7

A	B	A^B
0	0	0
1	1	0
0	1	1
1	0	1

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 so'rov qiymatlari shartga javob bersa, TRUE
AND	VA bilan ajratilgan barcha shartlar TRUE bo`lsa, TRUE
ANY	So'rov qiymatlaridan birortasi shartga javob bersa, TRUE
BETWEEN	Operand taqqoslash oralig`ida bo`lsa, TRUE
EXISTS	So'rov bir yoki bir nechta yozuvni qaytarsa, TRUE

IN	Operand ifodalar ro‘yxatidan biriga teng bo‘lsa, 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	So‘rov 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

CREATE TABLE ***DROP TABLE*** ***TRUNCATE TABLE***

Yangi jadval yaratish:

```
CREATE TABLE table_name (column1 datatype,
                         column2 datatype, ....);
```

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

DROP TABLE mavjud jadvalni bazadan o`chiradi.

```
DROP TABLE table_name;
```

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

```
TRUNCATE TABLE table_name;
```

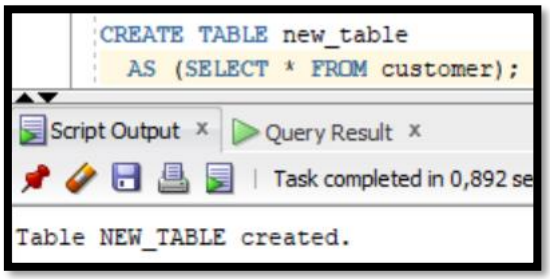
```
TRUNCATE TABLE t_name ⇔ DELETE FROM t_name
```

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);
```



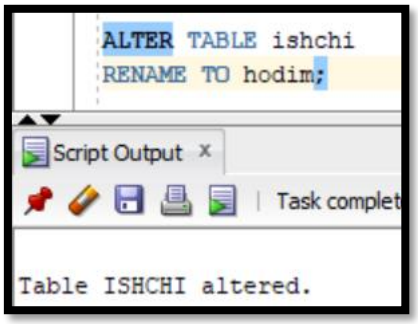
ALTER TABLE bayonoti

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.

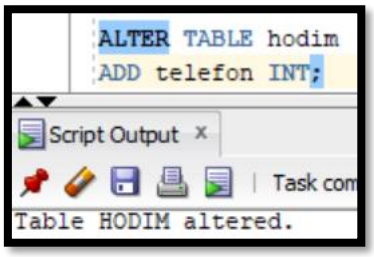
Jadval nomini o`zgartirish:

```
ALTER TABLE old_name
RENAME TO new_name;
```



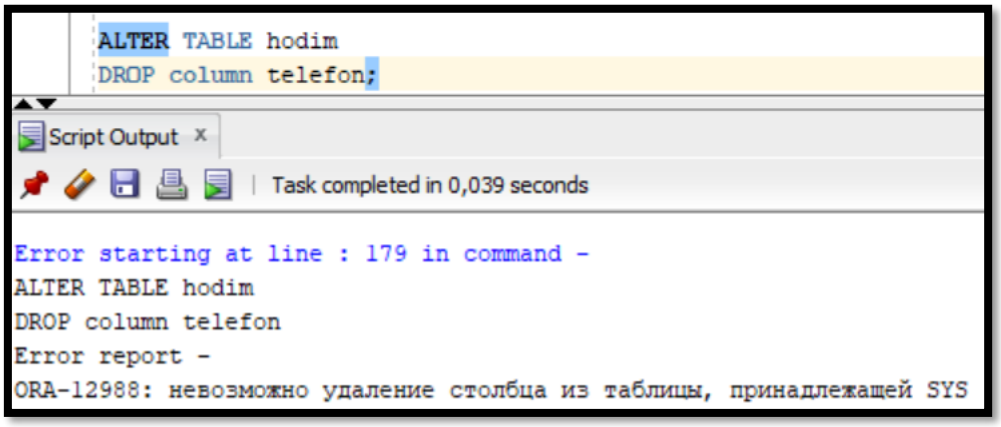
Jadvalga ustun qo`shish:

```
ALTER TABLE table_name
  ADD (column_1 datatype,
       ...);
```



Jadvaldagi ustunni o`chirish:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```



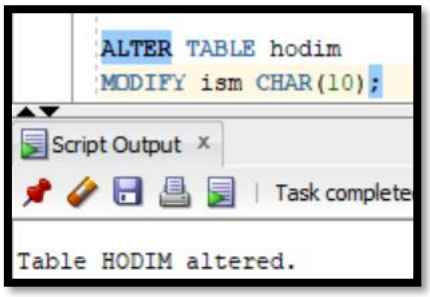
(Faqatgina SYS bazadan o’chirish mumkin emas,boshqasida m-n)

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 - Ustun NULL qiymatiga ega bo`lmasligini ta'minlaydi
- UNIQUE - Ustundagi barcha qiymatlar noyob bo`lishini ta'minlaydi
- PRIMARY KEY - NOT NULL va UNIQUE birikmasidir. Jadvaldagi har bir qatorni noyob tarzda aniqlaydi.

- FOREIGN KEY - Jadvallar orasidagi aloqalarni buzadigan harakatlarni oldini oladi.
- ChECK - Ustundagi qiymatlar ma'lum bir shartga mosligini ta'minlaydi.
- DEFAULT - Agar qiymat belgilanmagan bo`lsa, ustun uchun standart qiymatni o`rnatadi.
- CREATE INDEX - Ma'lumotlar bazasidan ma'lumotlarni juda tez yaratish va olish uchun foydalaniladi.

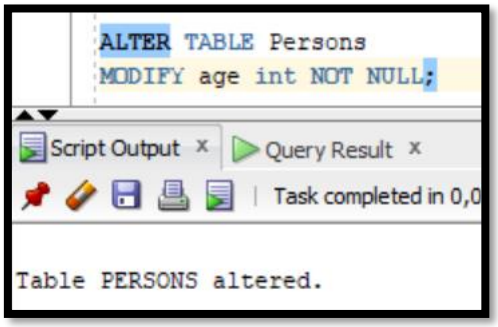
NOT NULL CHEKLOVI

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
);
```

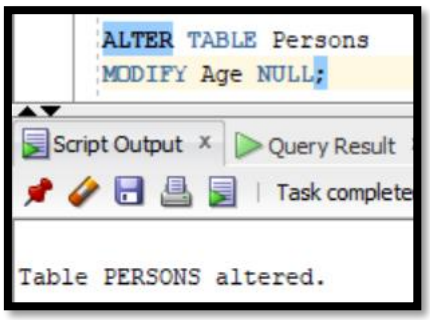
"Persons" jadvalining "Age" ustuniga NOT NULL cheklovini o'rnatish:

```
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 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 ChECK cheklovini 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.

	ID	ORDERDATE_1	ORDERDATE_2	ORDERDATE_3	ORDERDATE_4
1	1	27.03.23	27.03.23	27.03.23 08:52:55,000000000	27.03.23 08:52:55,881000000
2	2	27.03.23	27.03.23	27.03.23 08:53:04,000000000	27.03.23 08:53:04,362000000
3	3	27.03.23	27.03.23	27.03.23 08:53:10,000000000	27.03.23 08:53:10,006000000

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

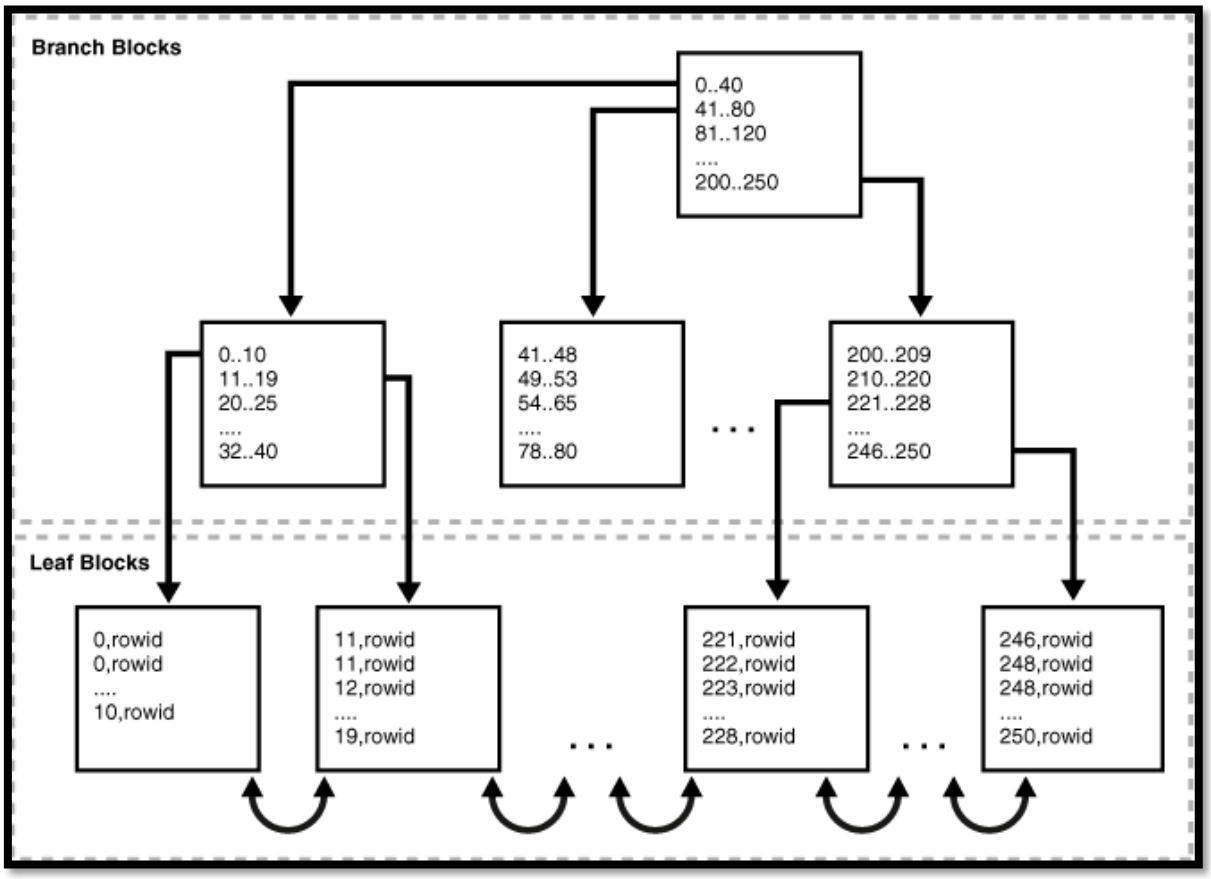
Indekslar ma’lumotlar bazasidan ma’lumotlarni tezroq, samarador olish uchun ishlatiladi. Foydalanuvchilar indekslarni ko`ra olmaydi, ular faqat qidiruv/so`rovlarni tezlashtirish uchun ishlatiladi.

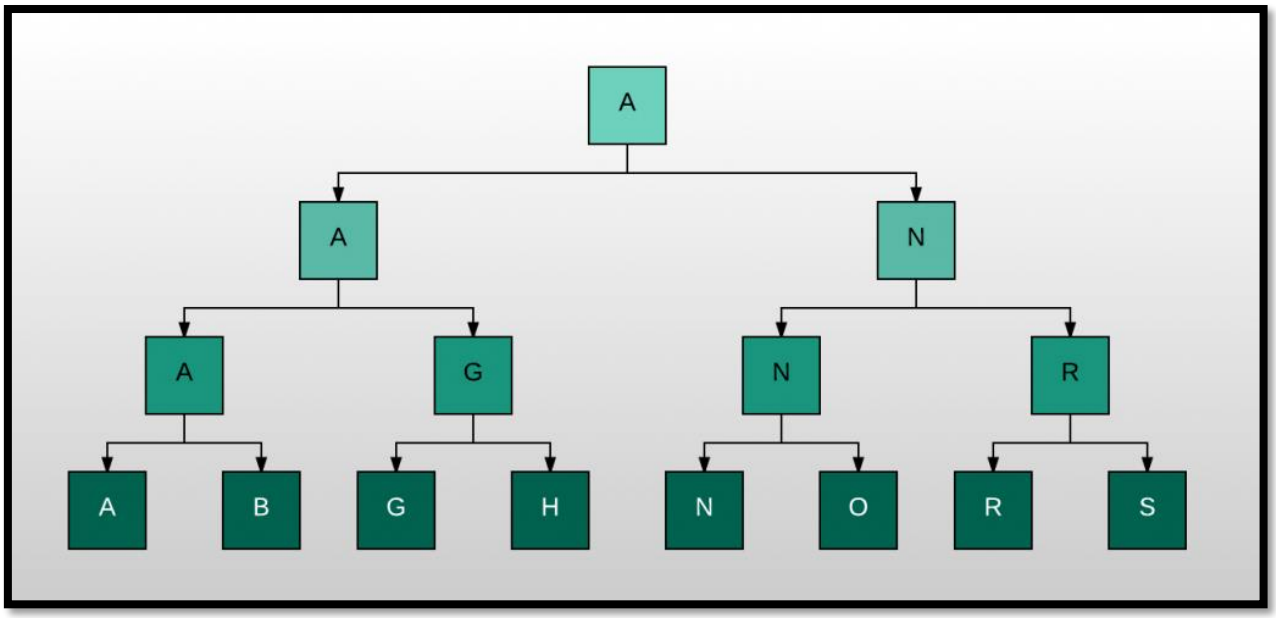
Eslatma: Faqatgina tez-tez qidiriladigan ustunlar bo`yicha indekslarni yarating.

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 9999999999999999999999999999
  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.

```
CACHE 20;
```

NOCACHE ketma-ketlik qiymatlarining hech biri xotirada saqlanmasligini anglatadi.

```
NOCACHE;
```

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:

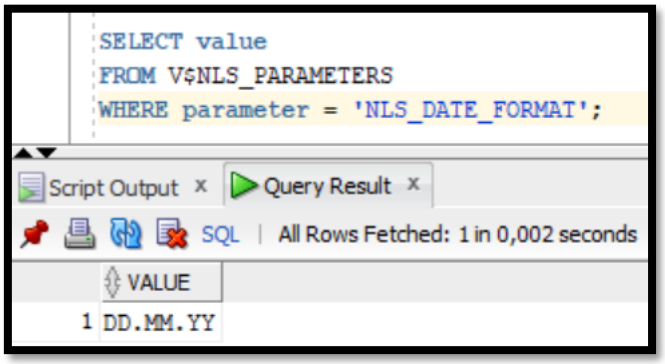
```
ALTER SEQUENCE seq_name
INCREMENT BY 124;


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:

```
SELECT sysdate
FROM dual;
```

Script Output × Query R

SQL | All Row

	SYSDATE
1	29.03.23

Sana formatini o’zgartirish:

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MM-YY';
```

Script Output × Query Result ×

Task completed in 0,028 seconds

```
Session altered.
```

Hozirgi sanani tekshiramiz:

```
SELECT sysdate
FROM dual;
```

Script Output × Que

SQL | All

	SYSDATE
1	29-03-23

DATES

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

Ma’lumotlaringizda faqat sana qismi bo‘lsa, so‘rovlaringiz 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;
```