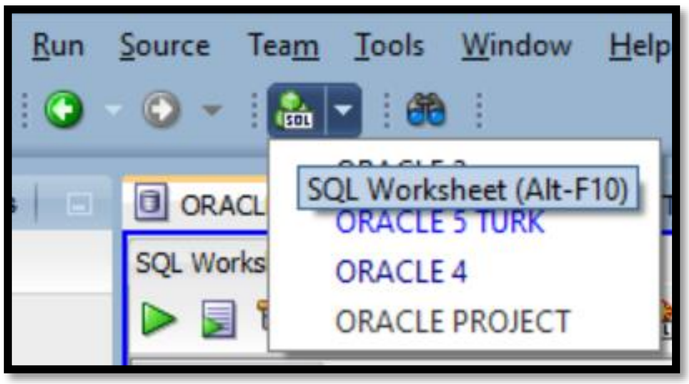
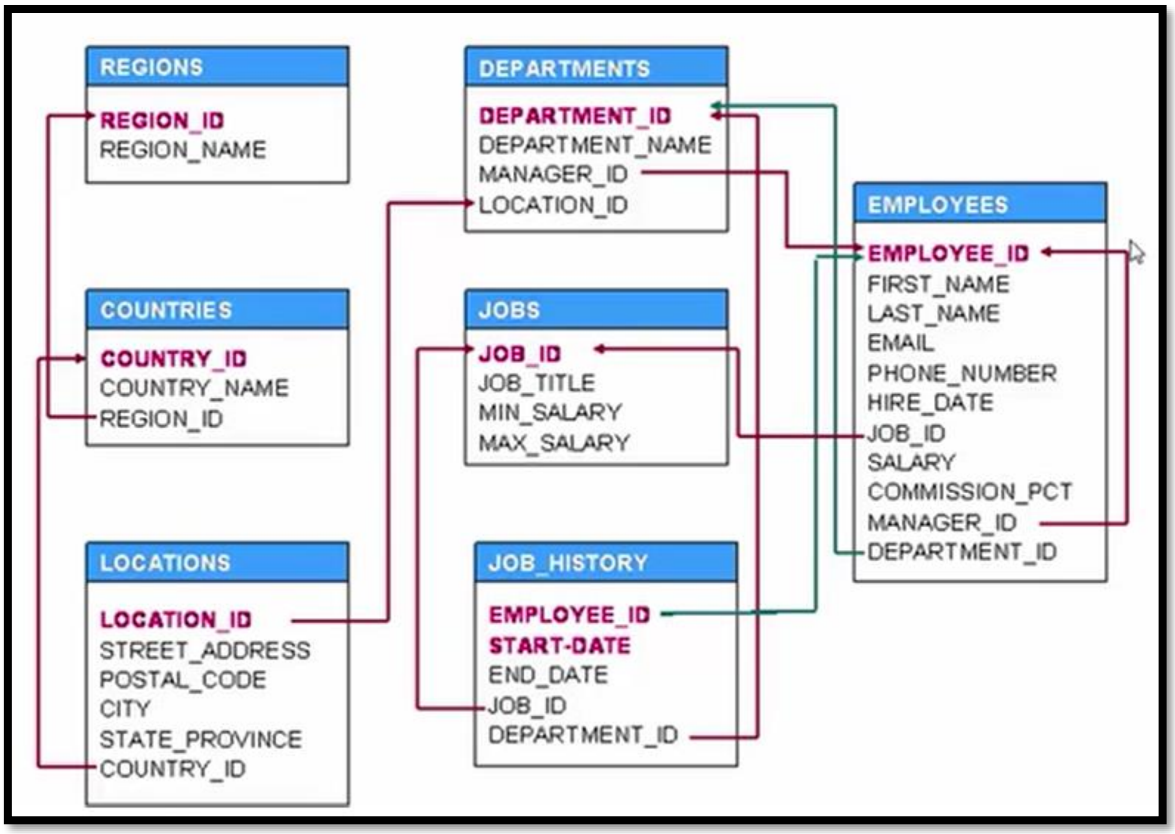
ORACLE®


Run
Source
Team
Tools
Window
Help
SQL Worksheet (Alt-F10)
ORACLE 5 TURK
ORACLE 4
ORACLE PROJECT
ORACL
SQL Works

1-qadam

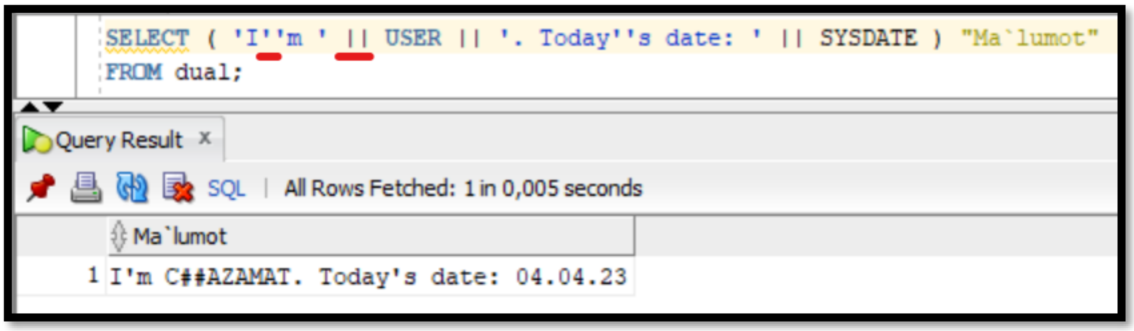
Ma’lumotlar modeli



Text chiqarish:

{ ‵ } bu belgi uchun 2ta qo`shtirnoq { ‶ } yonma yon yoziladi.

{ || } bu belgi ketma-ketliklarni qo`shib chiqar ma’nosini anglatadi.



```
SELECT ( 'I''m ' || USER || '. Today''s date: ' || SYSDATE ) "Ma`lumot"
FROM dual;
```

Query Result ×

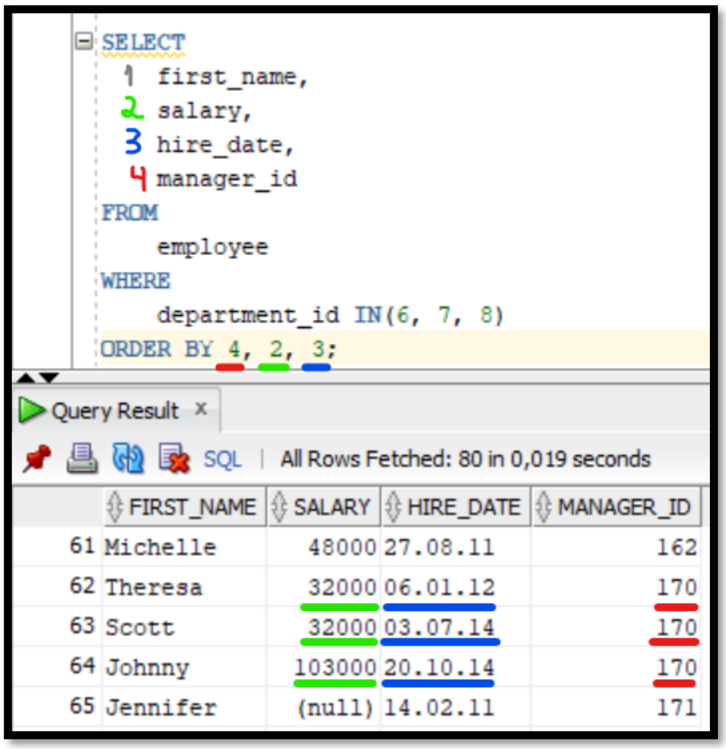
SQL | All Rows Fetched: 1 in 0,005 seconds

```
  Ma`lumot
1 I'm C##AZAMAT. Today's date: 04.04.23
```

ORDER BY bandi

Ustunlarni tartib nomeri bo`yicha saralash:

(Ketma-ketligi: 4-ustun, 2-ustun va 3-ustun bo`yicha tartiblanadi)



```
SELECT
  1 first_name,
  2 salary,
  3 hire_date,
  4 manager_id
FROM
    employee
WHERE
    department_id IN(6, 7, 8)
ORDER BY 4, 2, 3;
```

Query Result

All Rows Fetched: 80 in 0,019 seconds

	FIRST_NAME	SALARY	HIRE_DATE	MANAGER_ID
61	Michelle	48000	27.08.11	162
62	Theresa	32000	06.01.12	170
63	Scott	32000	03.07.14	170
64	Johnny	103000	20.10.14	170
65	Jennifer	(null)	14.02.11	171

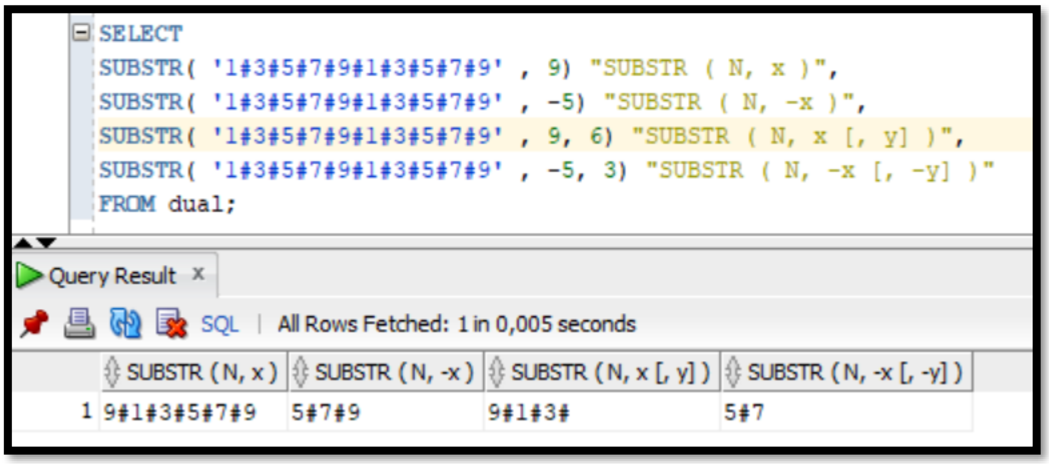
Substr funksiyasi

SUBSTR (N, x) → N ta belgili matndan x-belgidan boshlab yozadi.

SUBSTR (N, -x) → N ta belgili matndan oxiridan boshiga sanalib, x-belgidan boshlab yozadi.

SUBSTR (N, x [, y]) → x-belgidan boshlab, y ta belgi yozadi.

SUBSTR (N, -x [, y]) → oxiridan boshiga sanalib, x-belgidan boshlab, y ta belgi yozadi.



```
SELECT
SUBSTR( '1#3#5#7#9#1#3#5#7#9' , 9) "SUBSTR ( N, x )",
SUBSTR( '1#3#5#7#9#1#3#5#7#9' , -5) "SUBSTR ( N, -x )",
SUBSTR( '1#3#5#7#9#1#3#5#7#9' , 9, 6) "SUBSTR ( N, x [, y] )",
SUBSTR( '1#3#5#7#9#1#3#5#7#9' , -5, 3) "SUBSTR ( N, -x [, -y] )"
FROM dual;
```

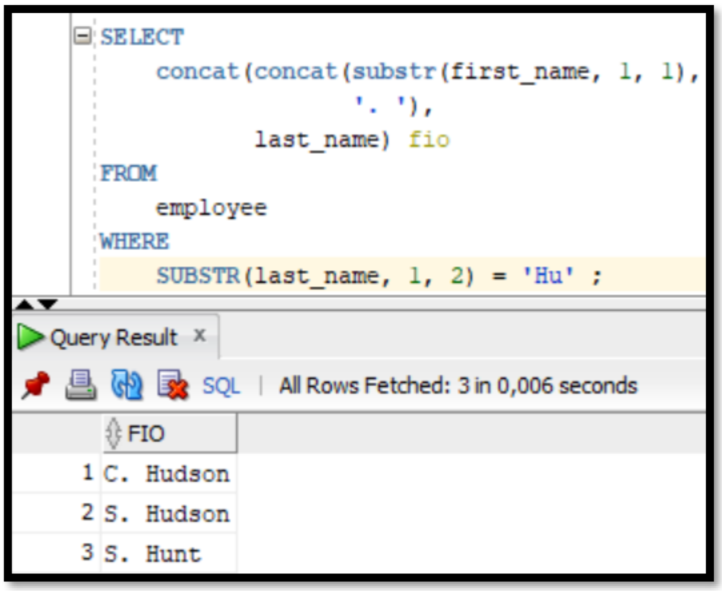
Query Result ×

All Rows Fetched: 1 in 0,005 seconds

	SUBSTR (N, x)	SUBSTR (N, -x)	SUBSTR (N, x [, y])	SUBSTR (N, -x [, -y])
1	9#1#3#5#7#9	5#7#9	9#1#3#	5#7

Ism familiyani qisqartirish:

*(SUBSTR*ni *WHERE* bandida ham ishlatsa bo`ladi *)*



```
SELECT
    concat(concat(substr(first_name, 1, 1),
                  '. '),
           last_name) fio
FROM
    employee
WHERE
    SUBSTR(last_name, 1, 2) = 'Hu' ;
```

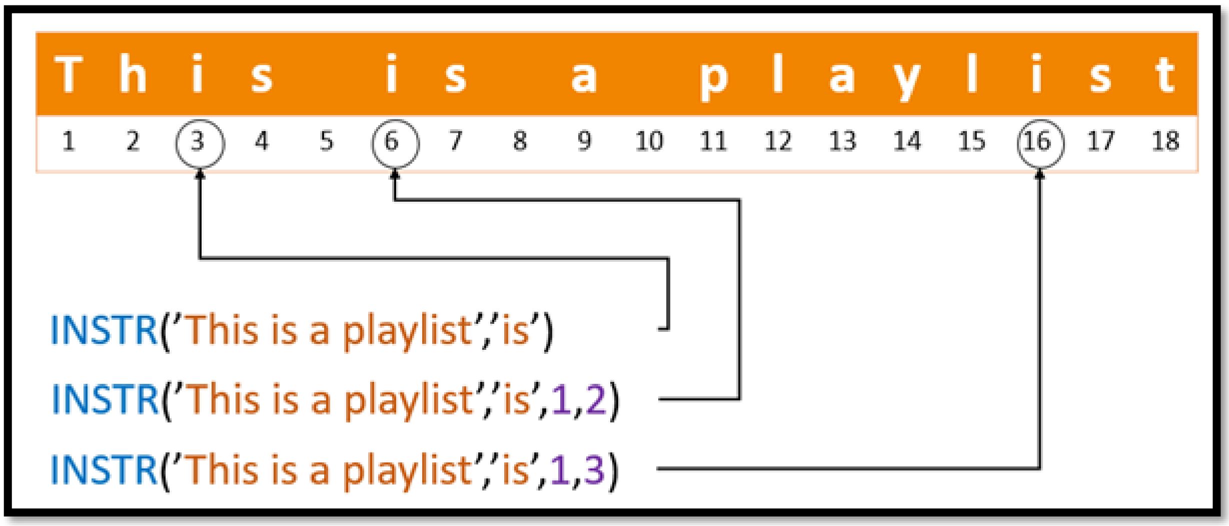
Query Result ×

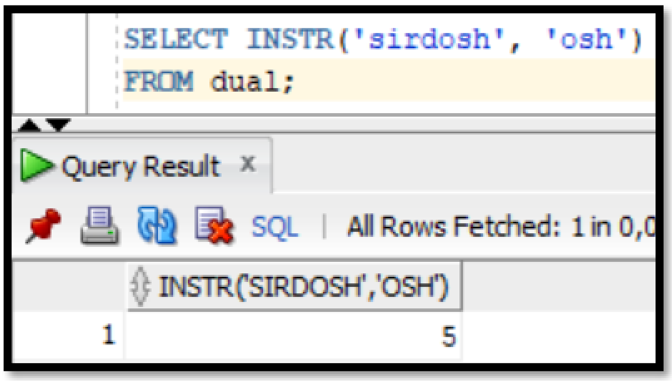
All Rows Fetched: 3 in 0,006 seconds

```
  FIO
1 C. Hudson
2 S. Hudson
3 S. Hunt
```

INSTR()

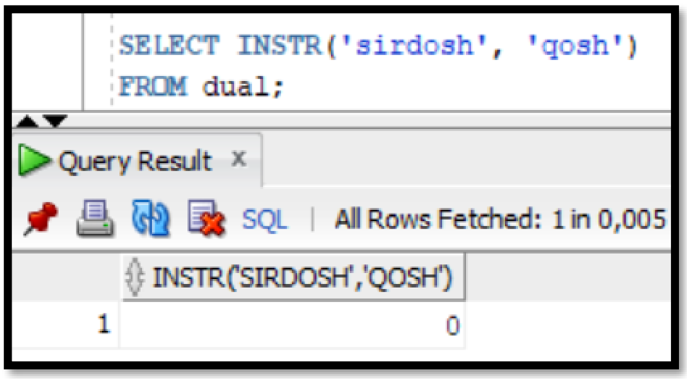
INSTR ⬄ berilgan matndan belgilangan belgilarni tartib nomerini ko`rsatadi.


T h i s i s a p l a y l i s t
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
INSTR('This is a playlist','is')
INSTR('This is a playlist','is',1,2)
INSTR('This is a playlist','is',1,3)


```
SELECT INSTR('sirdosh', 'osh')
FROM dual;
```
Query Result ×

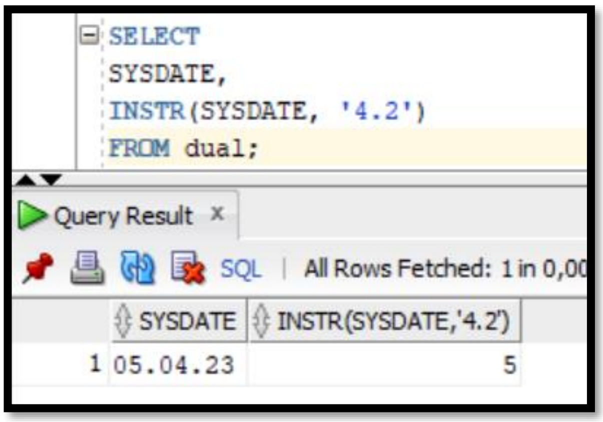
SQL | All Rows Fetched: 1 in 0,0

```
   INSTR('SIRDOSH','OSH')
1                        5
```


```
SELECT INSTR('sirdosh', 'qosh')
FROM dual;
```
Query Result ×

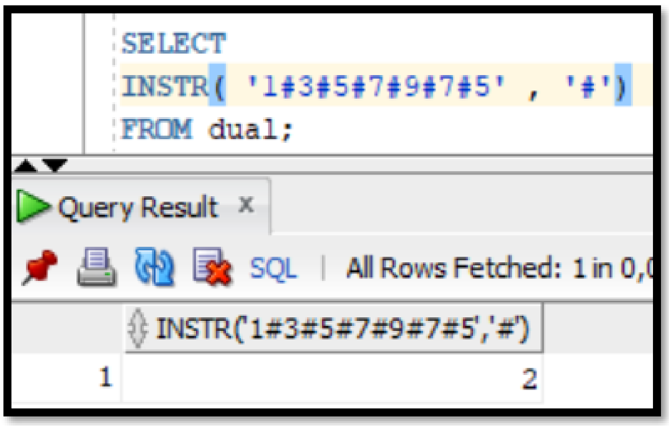
SQL | All Rows Fetched: 1 in 0,005

```
   INSTR('SIRDOSH','QOSH')
1                         0
```


```
SELECT
SYSDATE,
INSTR(SYSDATE, '4.2')
FROM dual;
```
Query Result ×

SQL | All Rows Fetched: 1 in 0,00

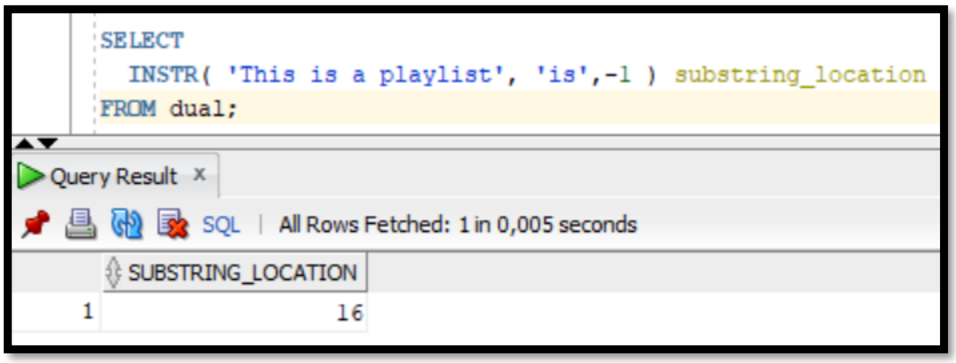
```
   SYSDATE    INSTR(SYSDATE,'4.2')
1  05.04.23                      5
```


```
SELECT
INSTR( '1#3#5#7#9#7#5' , '#')
FROM dual;
```
Query Result ×

SQL | All Rows Fetched: 1 in 0,0

```
   INSTR('1#3#5#7#9#7#5','#')
1                            2
```

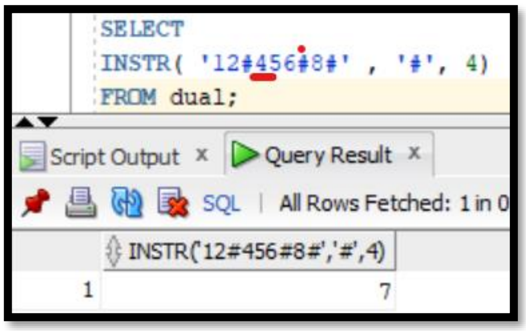
Oxiridan boshiga qarab birinchi takrorlanishni qidirish:


```
SELECT
  INSTR( 'This is a playlist', 'is',-1 ) substring_location
FROM dual;
```
Query Result ×

SQL | All Rows Fetched: 1 in 0,005 seconds

```
   SUBSTRING_LOCATION
1                  16
```

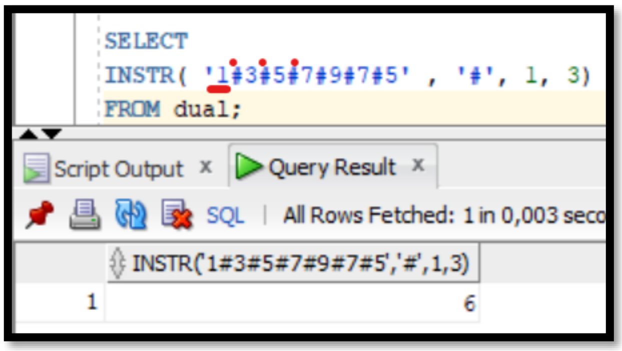
🡻 4- belgidan keyin keladigan # ni tartib nomerini topish:


```
SELECT
INSTR( '12#456#8#' , '#', 4)
FROM dual;
```
Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0

```
  INSTR('12#456#8#','#',4)
1                        7
```

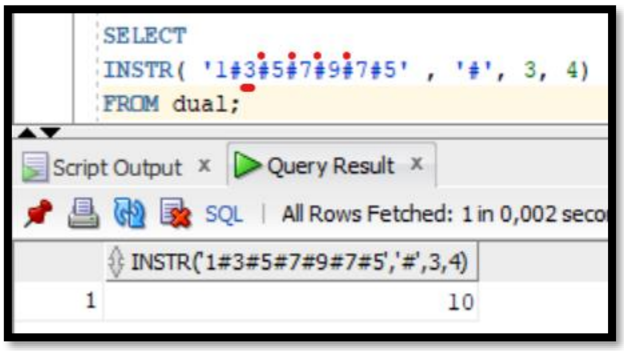
1-belgidan keyin keladigan 3- marta takrorlanadigan # ni tartib nomerini topish:


```
SELECT
INSTR( '1#3#5#7#9#7#5' , '#', 1, 3)
FROM dual;
```
Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0,003 seco

```
  INSTR('1#3#5#7#9#7#5','#',1,3)
1                              6
```

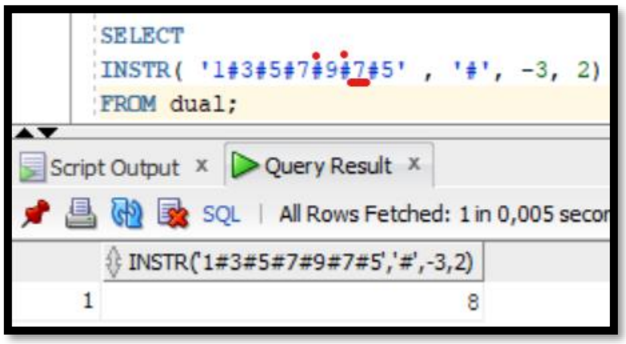
3-belgidan keyin keladigan 4-marta takrorlanadigan # ni tartib nomerini topish:


```
SELECT
INSTR( '1#3#5#7#9#7#5' , '#', 3, 4)
FROM dual;
```
Script Output × | Query Result ×

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

```
  INSTR('1#3#5#7#9#7#5','#',3,4)
1                             10
```

(-3)-belgidan boshlab, boshiga qarab keladigan 2-marta takrorlanadigan # ni tartib nomerini topish:

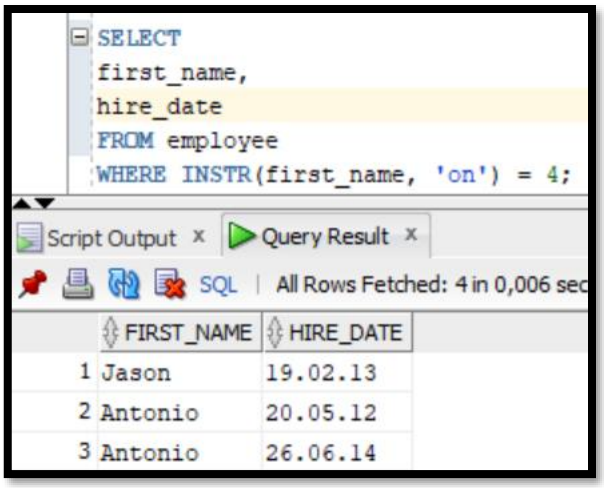

```
SELECT
INSTR( '1#3#5#7#9#7#5' , '#', -3, 2)
FROM dual;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0,005 secon

```
   INSTR('1#3#5#7#9#7#5','#',-3,2)
1                                8
```

INSTR ni *WHERE* bandida ishlatilishi:


```
SELECT
first_name,
hire_date
FROM employee
WHERE INSTR(first_name, 'on') = 4;
```

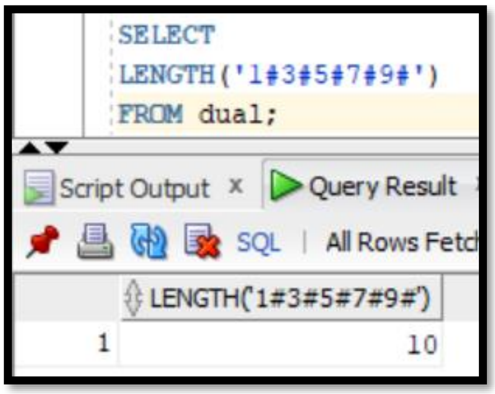
Script Output × | Query Result ×

SQL | All Rows Fetched: 4 in 0,006 sec

```
   FIRST_NAME  HIRE_DATE
1  Jason       19.02.13
2  Antonio     20.05.12
3  Antonio     26.06.14
```

LENGTH() funksiyasi

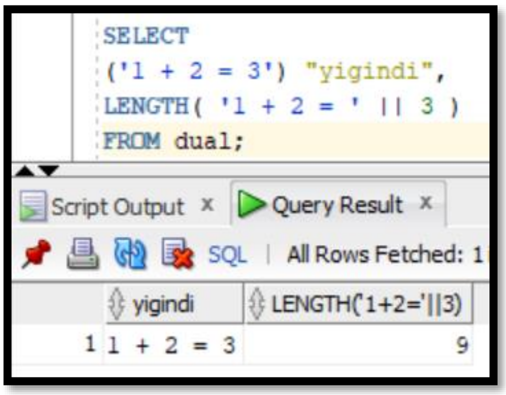
Belgilar sonini aniqlash:


```
SELECT
LENGTH('1#3#5#7#9#')
FROM dual;
```

Script Output × | Query Result

SQL | All Rows Fetd

```
   LENGTH('1#3#5#7#9#')
1                    10
```

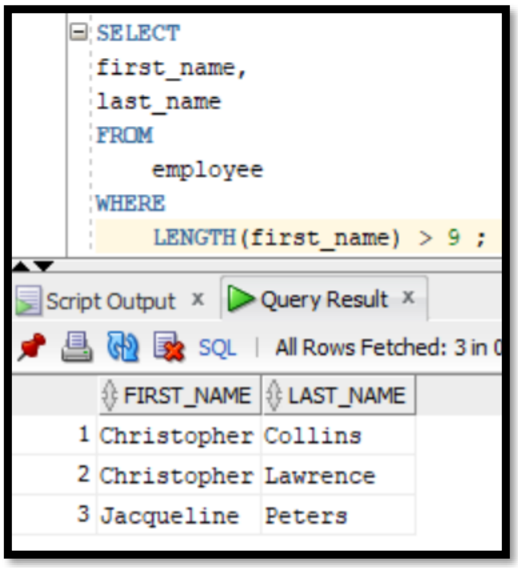

```
SELECT
('1 + 2 = 3') "yigindi",
LENGTH( '1 + 2 = ' || 3 )
FROM dual;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 1

```
   yigindi    LENGTH('1+2='||3)
1  1 + 2 = 3                  9
```

Where bandida ishlatilishi:



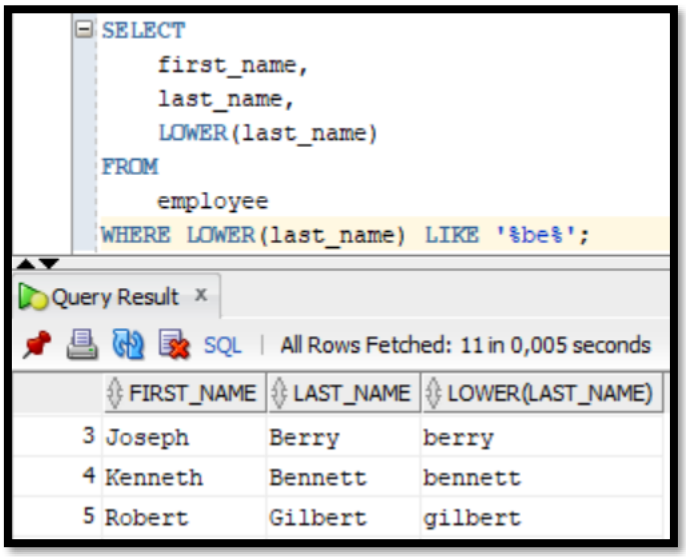
```
SELECT
first_name,
last_name
FROM
    employee
WHERE
    LENGTH(first_name) > 9 ;
```

```
  FIRST_NAME  LAST_NAME
1 Christopher Collins
2 Christopher Lawrence
3 Jacqueline  Peters
```

LOWER() Funksiyasi

Lower → kichik harflarga o`tkazadi.

Where bandida ishlatilishi:

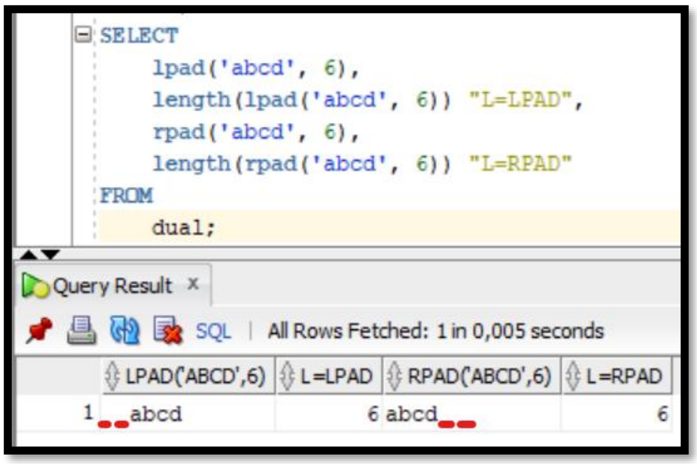


```
SELECT
    first_name,
    last_name,
    LOWER(last_name)
FROM
    employee
WHERE LOWER(last_name) LIKE '%be%';
```

```
  FIRST_NAME LAST_NAME LOWER(LAST_NAME)
3 Joseph     Berry     berry
4 Kenneth    Bennett   bennett
5 Robert     Gilbert   gilbert
```

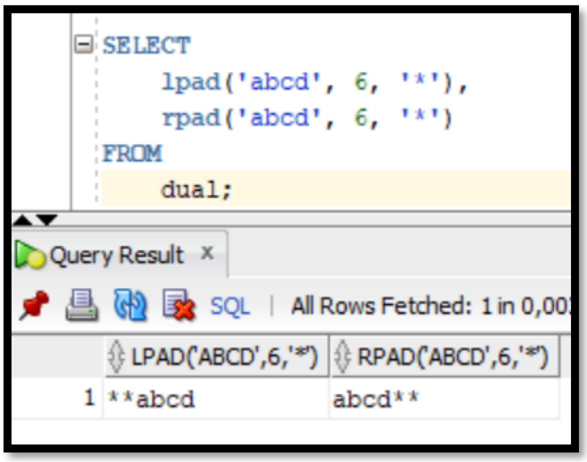
LPAD, RPAD

Lpad(c1, x1 [, y1])



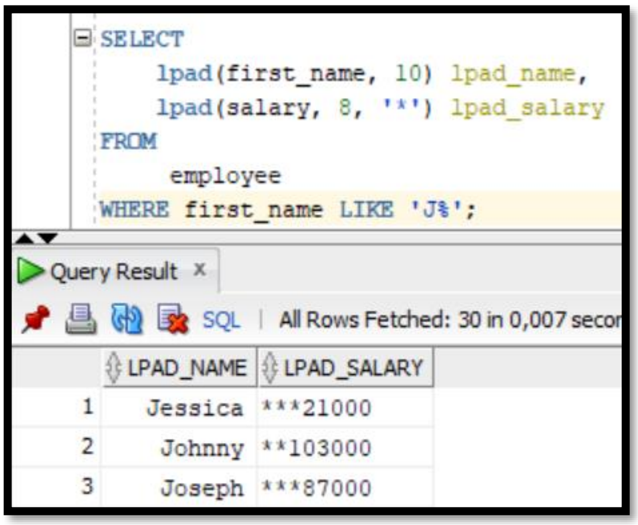
```sql
SELECT
    lpad('abcd', 6),
    length(lpad('abcd', 6)) "L=LPAD",
    rpad('abcd', 6),
    length(rpad('abcd', 6)) "L=RPAD"
FROM
    dual;
```

```
Query Result ×
SQL | All Rows Fetched: 1 in 0,005 seconds

  LPAD('ABCD',6)  L=LPAD  RPAD('ABCD',6)  L=RPAD
1   abcd               6  abcd                 6
```



```sql
SELECT
    lpad('abcd', 6, '*'),
    rpad('abcd', 6, '*')
FROM
    dual;
```

```
Query Result ×
SQL | All Rows Fetched: 1 in 0,00

  LPAD('ABCD',6,'*')  RPAD('ABCD',6,'*')
1 **abcd              abcd**
```

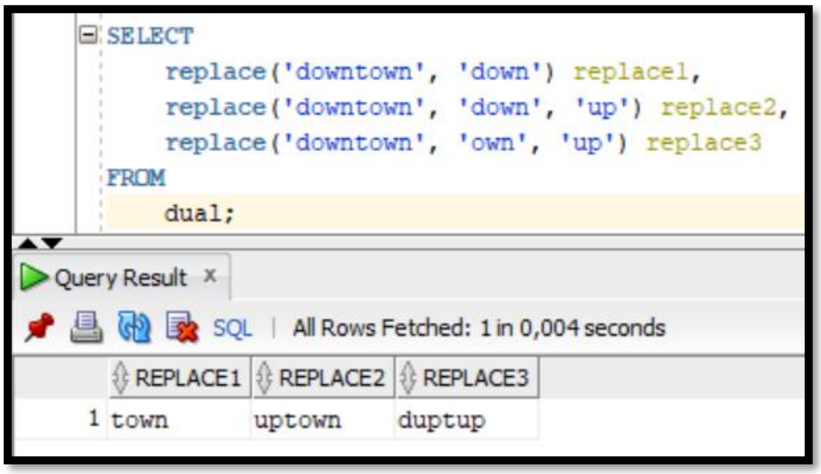


```sql
SELECT
    lpad(first_name, 10) lpad_name,
    lpad(salary, 8, '*') lpad_salary
FROM
     employee
WHERE first_name LIKE 'J%';
```

```
Query Result ×
SQL | All Rows Fetched: 30 in 0,007 secor

  LPAD_NAME   LPAD_SALARY
1    Jessica  ***21000
2     Johnny  **103000
3     Joseph  ***87000
```

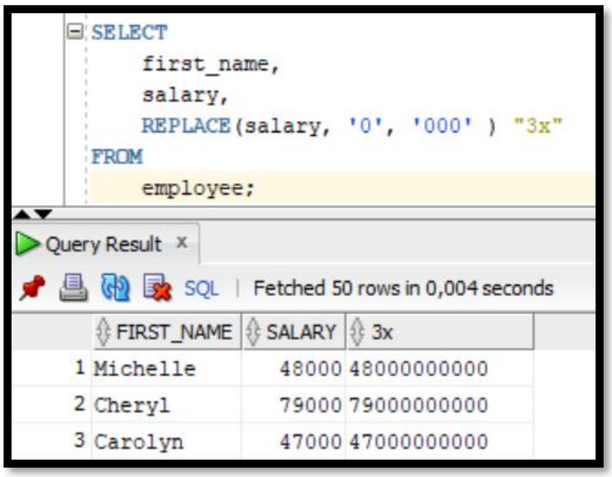
REPLACE

REPLACE(c1, c2 [, c3]) → satrdagi belgilangan pastki qatorning barcha takrorlanishini boshqasiga almashtiradi.



```sql
SELECT
    replace('downtown', 'down') replace1,
    replace('downtown', 'down', 'up') replace2,
    replace('downtown', 'own', 'up') replace3
FROM
    dual;
```

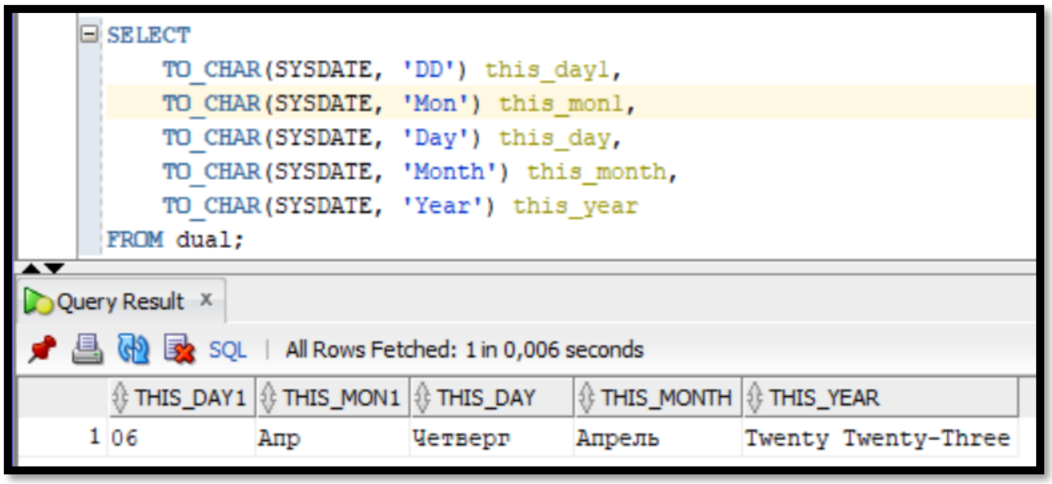
```
Query Result ×
SQL | All Rows Fetched: 1 in 0,004 seconds

  REPLACE1  REPLACE2  REPLACE3
1 town      uptown    duptup
```



```sql
SELECT
    first_name,
    salary,
    REPLACE(salary, '0', '000' ) "3x"
FROM
    employee;
```

```
Query Result ×
SQL | Fetched 50 rows in 0,004 seconds

  FIRST_NAME  SALARY  3x
1 Michelle     48000  48000000000
2 Cheryl       79000  79000000000
3 Carolyn      47000  47000000000
```

TO_ChAR

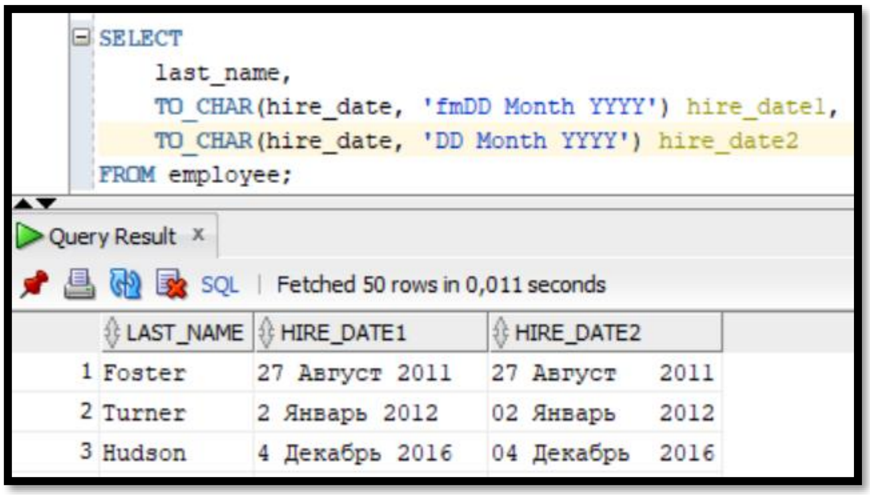


```
SELECT
    TO_CHAR(SYSDATE, 'DD') this_day1,
    TO_CHAR(SYSDATE, 'Mon') this_mon1,
    TO_CHAR(SYSDATE, 'Day') this_day,
    TO_CHAR(SYSDATE, 'Month') this_month,
    TO_CHAR(SYSDATE, 'Year') this_year
FROM dual;
```

Query Result

SQL | All Rows Fetched: 1 in 0,006 seconds

	THIS_DAY1	THIS_MON1	THIS_DAY	THIS_MONTH	THIS_YEAR
1	06	Апр	Четверг	Апрель	Twenty Twenty-Three



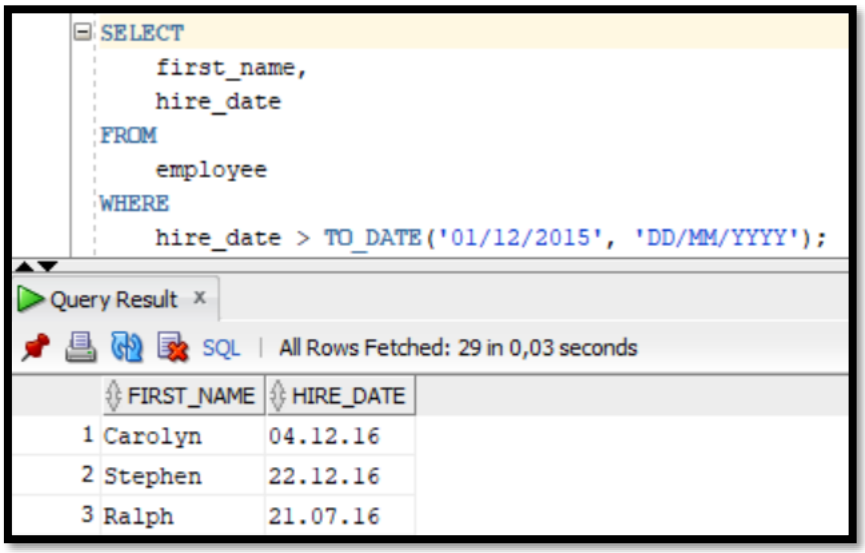
```
SELECT
    last_name,
    TO_CHAR(hire_date, 'fmDD Month YYYY') hire_date1,
    TO_CHAR(hire_date, 'DD Month YYYY') hire_date2
FROM employee;
```

Query Result

SQL | Fetched 50 rows in 0,011 seconds

	LAST_NAME	HIRE_DATE1	HIRE_DATE2
1	Foster	27 Август 2011	27 Август 2011
2	Turner	2 Январь 2012	02 Январь 2012
3	Hudson	4 Декабрь 2016	04 Декабрь 2016

TO_DATE



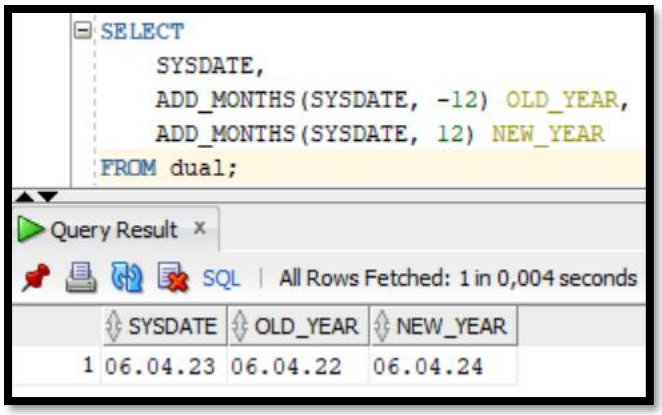
```
SELECT
    first_name,
    hire_date
FROM
    employee
WHERE
    hire_date > TO_DATE('01/12/2015', 'DD/MM/YYYY');
```

Query Result

SQL | All Rows Fetched: 29 in 0,03 seconds

	FIRST_NAME	HIRE_DATE
1	Carolyn	04.12.16
2	Stephen	22.12.16
3	Ralph	21.07.16

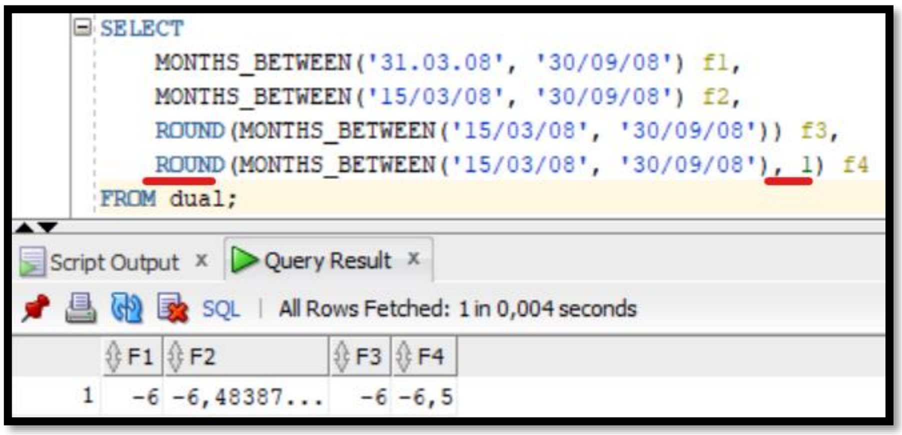
ADD_MONTHS



```
SELECT
    SYSDATE,
    ADD_MONTHS(SYSDATE, -12) OLD_YEAR,
    ADD_MONTHS(SYSDATE, 12) NEW_YEAR
FROM dual;
```

```
  SYSDATE   OLD_YEAR  NEW_YEAR
1 06.04.23  06.04.22  06.04.24
```

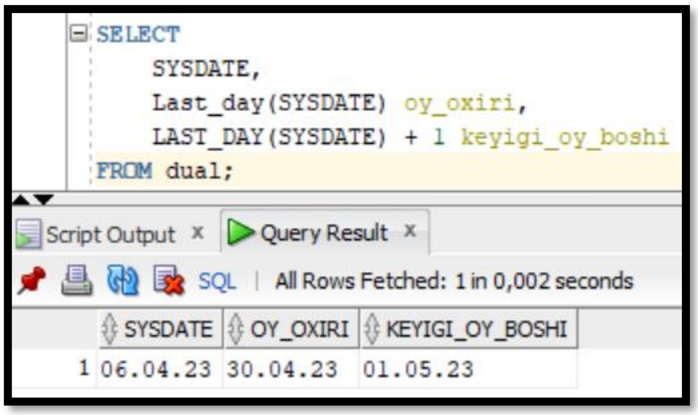
MONTHS_BETWEEN



```
SELECT
    MONTHS_BETWEEN('31.03.08', '30/09/08') f1,
    MONTHS_BETWEEN('15/03/08', '30/09/08') f2,
    ROUND(MONTHS_BETWEEN('15/03/08', '30/09/08')) f3,
    ROUND(MONTHS_BETWEEN('15/03/08', '30/09/08'), 1) f4
FROM dual;
```

```
  F1  F2          F3  F4
1 -6  -6,48387...  -6 -6,5
```

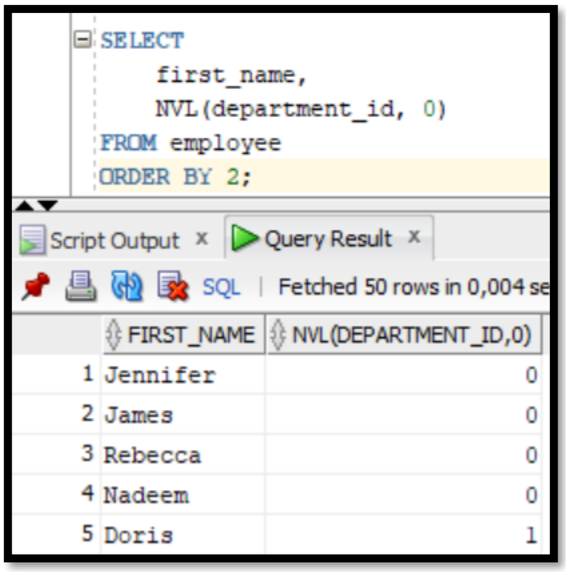
LAST_DAY



```
SELECT
    SYSDATE,
    Last_day(SYSDATE) oy_oxiri,
    LAST_DAY(SYSDATE) + 1 keyigi_oy_boshi
FROM dual;
```

```
  SYSDATE   OY_OXIRI  KEYIGI_OY_BOSHI
1 06.04.23  30.04.23  01.05.23
```

NVL

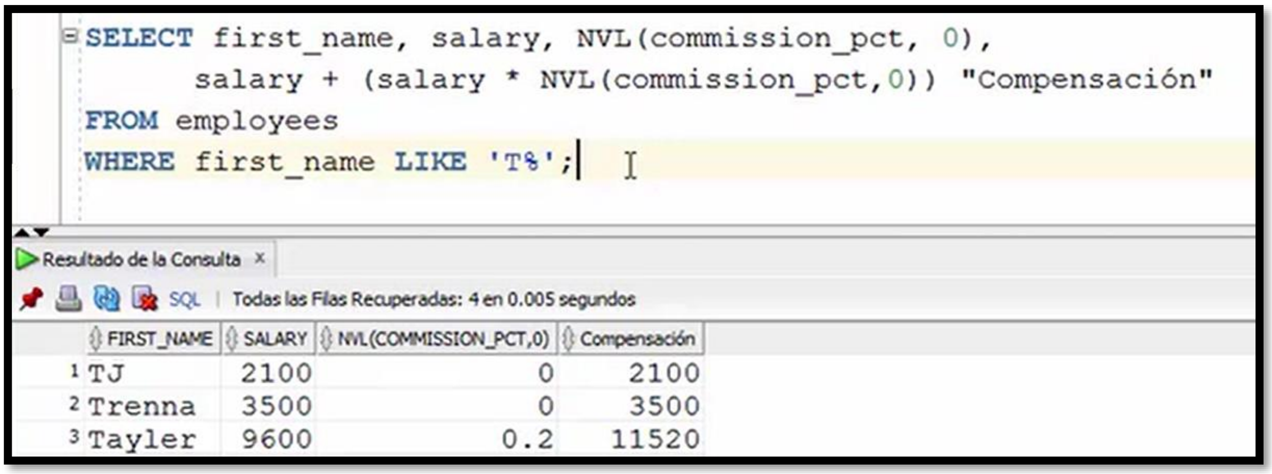


```
SELECT
    first_name,
    NVL(department_id, 0)
FROM employee
ORDER BY 2;
```

Script Output × | Query Result ×

SQL | Fetched 50 rows in 0,004 se

```
  FIRST_NAME  NVL(DEPARTMENT_ID,0)
1 Jennifer                       0
2 James                          0
3 Rebecca                        0
4 Nadeem                         0
5 Doris                          1
```



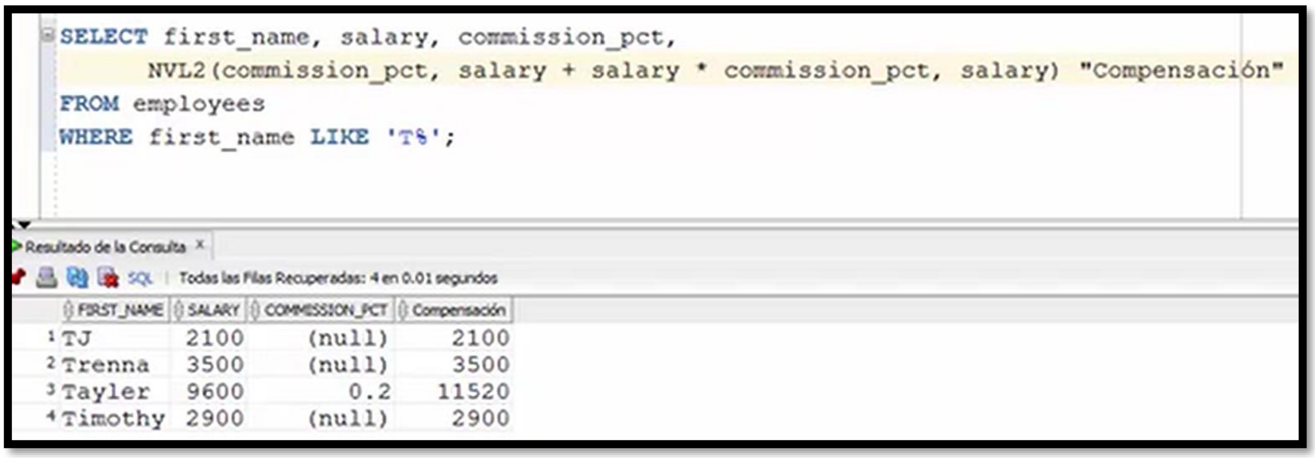
```
SELECT first_name, salary, NVL(commission_pct, 0),
       salary + (salary * NVL(commission_pct,0)) "Compensación"
FROM employees
WHERE first_name LIKE 'T%';
```

Resultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 4 en 0.005 segundos

```
  FIRST_NAME SALARY NVL(COMMISSION_PCT,0) Compensación
1 TJ          2100                      0         2100
2 Trenna      3500                      0         3500
3 Tayler      9600                    0.2        11520
```

NVL2



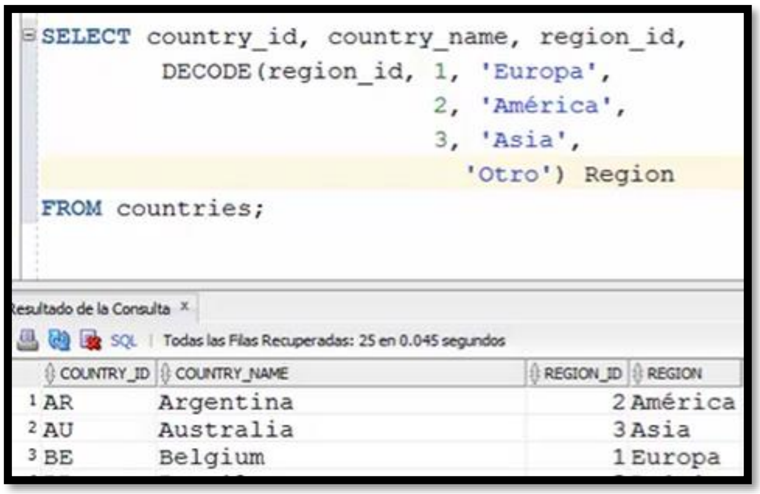
```
SELECT first_name, salary, commission_pct,
       NVL2(commission_pct, salary + salary * commission_pct, salary) "Compensación"
FROM employees
WHERE first_name LIKE 'T%';
```

Resultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 4 en 0.01 segundos

```
  FIRST_NAME SALARY COMMISSION_PCT Compensación
1 TJ          2100         (null)         2100
2 Trenna      3500         (null)         3500
3 Tayler      9600            0.2        11520
4 Timothy     2900         (null)         2900
```

DECODE()



```
SELECT country_id, country_name, region_id,
        DECODE(region_id, 1, 'Europa',
                          2, 'América',
                          3, 'Asia',
                            'Otro') Region
FROM countries;
```

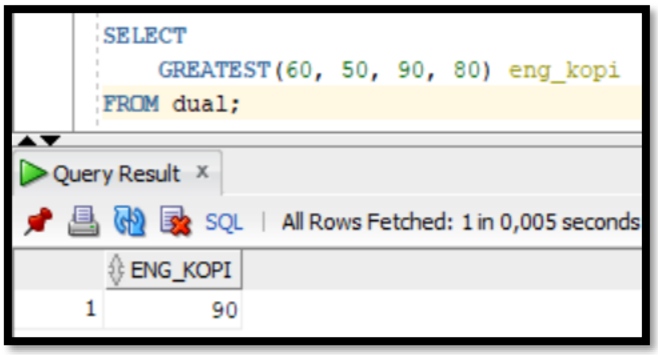
Resultado de la Consulta ×

SQL | Todas las Filas Recuperadas: 25 en 0.045 segundos

```
  COUNTRY_ID  COUNTRY_NAME   REGION_ID  REGION
1 AR          Argentina              2  América
2 AU          Australia              3  Asia
3 BE          Belgium                1  Europa
```

GREATEST()

Eng kattasini chiqaradi:

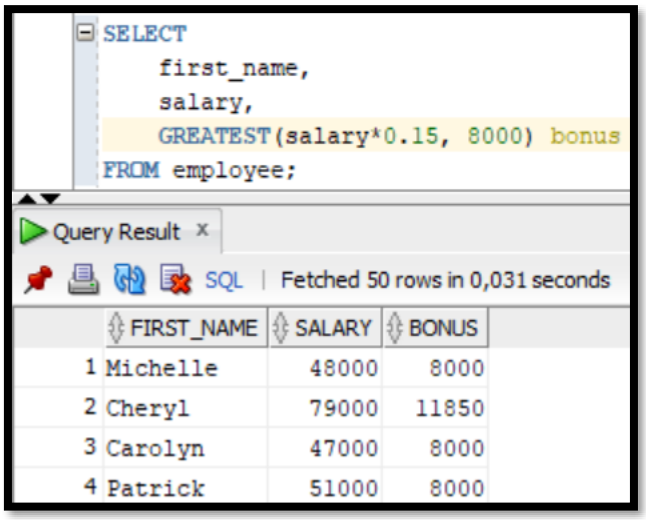


```
SELECT
    GREATEST(60, 50, 90, 80) eng_kopi
FROM dual;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0,005 seconds

```
  ENG_KOPI
1       90
```



```
SELECT
    first_name,
    salary,
    GREATEST(salary*0.15, 8000) bonus
FROM employee;
```

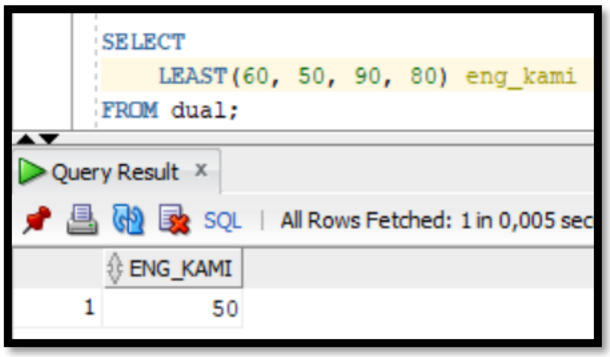
Query Result ×

SQL | Fetched 50 rows in 0,031 seconds

```
  FIRST_NAME  SALARY  BONUS
1 Michelle     48000   8000
2 Cheryl       79000  11850
3 Carolyn      47000   8000
4 Patrick      51000   8000
```

LEAST()

Eng kamini chiqaradi:

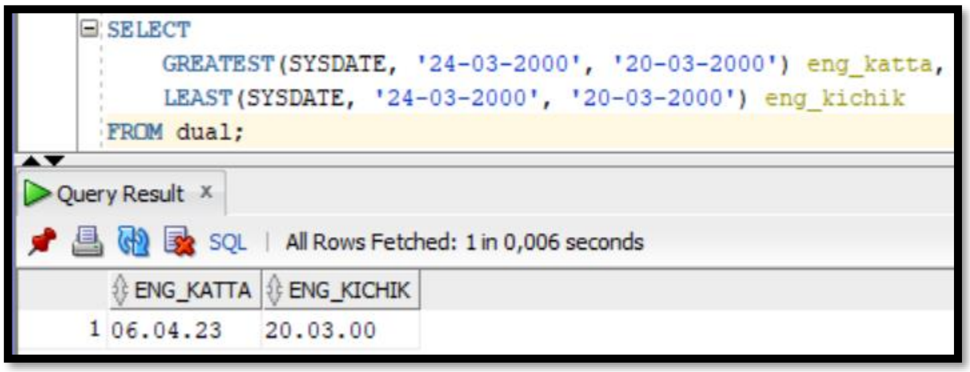


```
SELECT
    LEAST(60, 50, 90, 80) eng_kami
FROM dual;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0,005 sec

```
     ENG_KAMI
1          50
```



```
SELECT
    GREATEST(SYSDATE, '24-03-2000', '20-03-2000') eng_katta,
    LEAST(SYSDATE, '24-03-2000', '20-03-2000') eng_kichik
FROM dual;
```

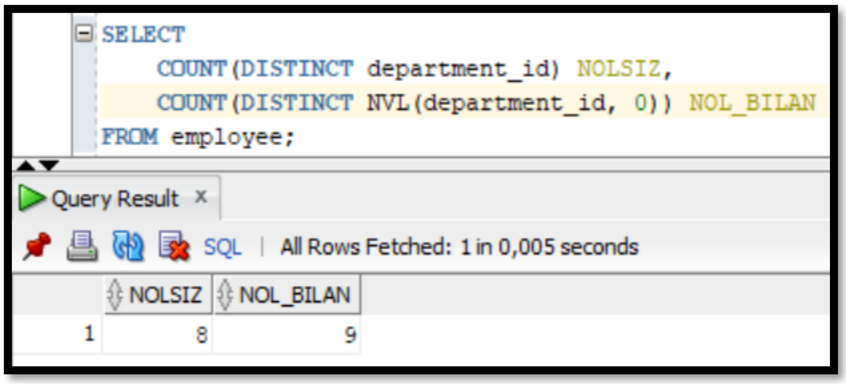
Query Result ×

SQL | All Rows Fetched: 1 in 0,006 seconds

```
  ENG_KATTA  ENG_KICHIK
1 06.04.23   20.03.00
```

Group By

COUNT()



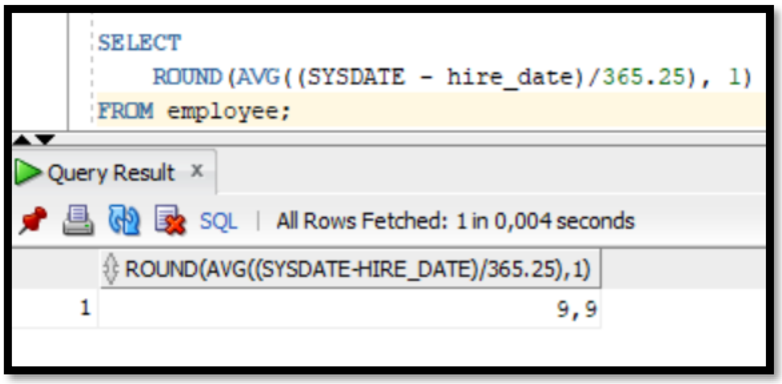
```
SELECT
    COUNT(DISTINCT department_id) NOLSIZ,
    COUNT(DISTINCT NVL(department_id, 0)) NOL_BILAN
FROM employee;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0,005 seconds

```
    NOLSIZ  NOL_BILAN
1        8          9
```

AVG()



```
SELECT
    ROUND(AVG((SYSDATE - hire_date)/365.25), 1)
FROM employee;
```

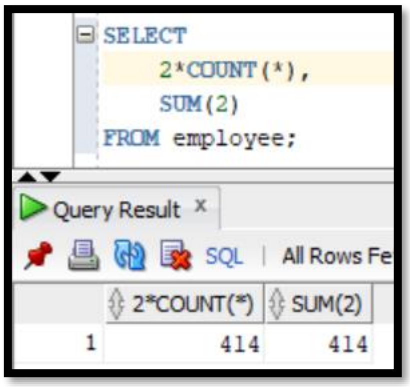
Query Result ×

SQL | All Rows Fetched: 1 in 0,004 seconds

```
  ROUND(AVG((SYSDATE-HIRE_DATE)/365.25),1)
1                                      9,9
```

SUM()

Ikkovi sinonim:



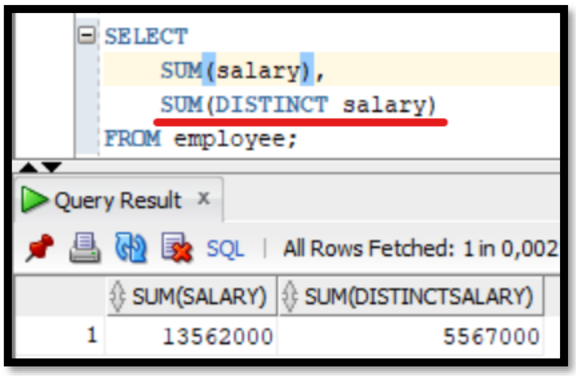
```
SELECT
    2*COUNT(*),
    SUM(2)
FROM employee;
```

Query Result ×

SQL | All Rows Fe

```
  2*COUNT(*)  SUM(2)
1        414     414
```

Sum(Distinct salary)

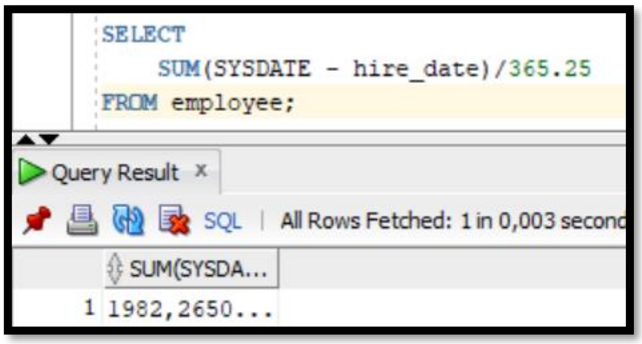


```
SELECT
    SUM(salary),
    SUM(DISTINCT salary)
FROM employee;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0,002

```
  SUM(SALARY)  SUM(DISTINCTSALARY)
1    13562000              5567000
```



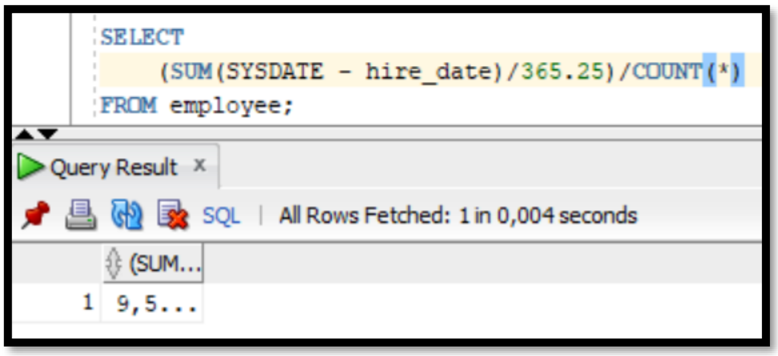
```
SELECT
    SUM(SYSDATE - hire_date)/365.25
FROM employee;
```

Query Result ×

SQL | All Rows Fetched: 1 in 0,003 second

```
  SUM(SYSDA...
1 1982,2650...
```

Hodimlarni o`rtacha ishlash yili:



```
SELECT
    (SUM(SYSDATE - hire_date)/365.25)/COUNT(*)
FROM employee;
```

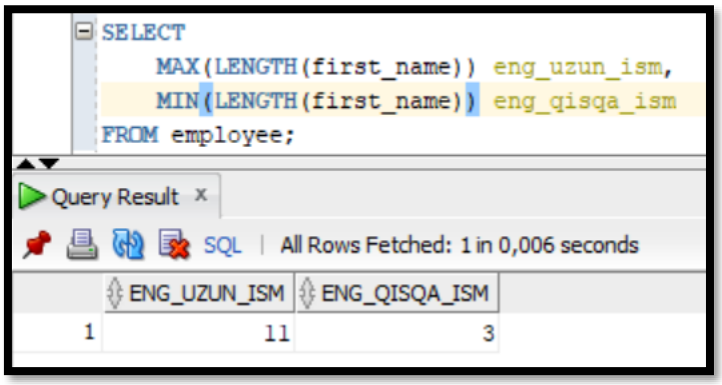
Query Result ×

SQL | All Rows Fetched: 1 in 0,004 seconds

```
  (SUM...
1 9,5...
```

MIN(), MAX()

Max va Min uzunlikdagi ismlarni topish:



```
SELECT
    MAX(LENGTH(first_name)) eng_uzun_ism,
    MIN(LENGTH(first_name)) eng_qisqa_ism
FROM employee;
```

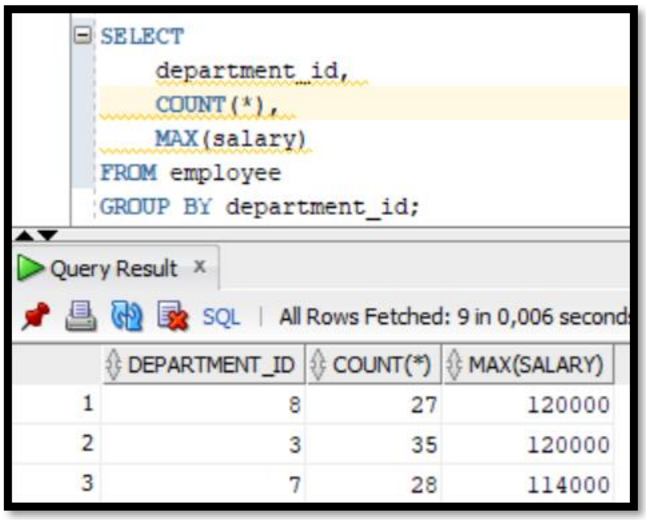
Query Result ×

SQL | All Rows Fetched: 1 in 0,006 seconds

```
  ENG_UZUN_ISM ENG_QISQA_ISM
1           11             3
```

GROUP BY

Har bir departament bo`yicha maksimal *salary*ni chiqarish:

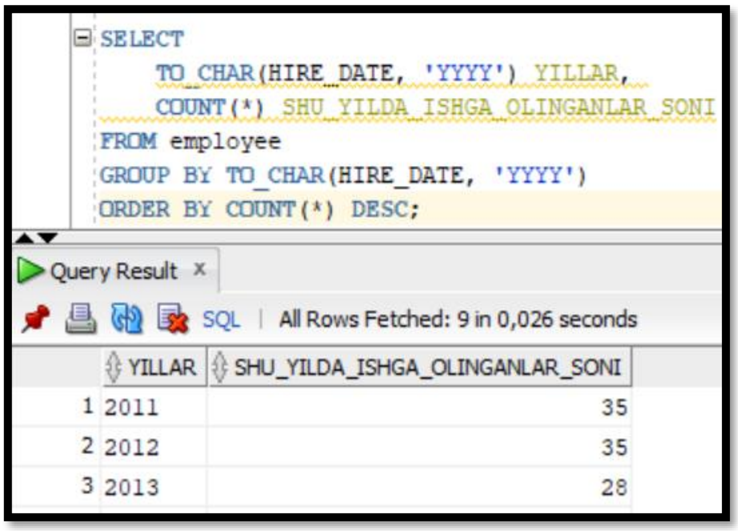


```
SELECT
    department_id,
    COUNT(*),
    MAX(salary)
FROM employee
GROUP BY department_id;
```

All Rows Fetched: 9 in 0,006 second

```
   DEPARTMENT_ID  COUNT(*)  MAX(SALARY)
1              8        27       120000
2              3        35       120000
3              7        28       114000
```

Yil bo`yicha ishga olinganlar sonini topish:

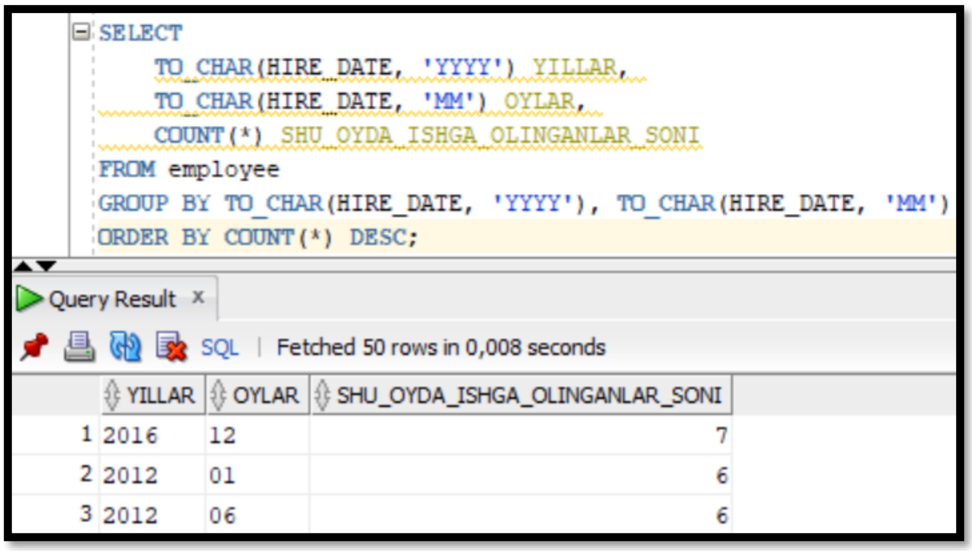


```
SELECT
    TO_CHAR(HIRE_DATE, 'YYYY') YILLAR,
    COUNT(*) SHU_YILDA_ISHGA_OLINGANLAR_SONI
FROM employee
GROUP BY TO_CHAR(HIRE_DATE, 'YYYY')
ORDER BY COUNT(*) DESC;
```

All Rows Fetched: 9 in 0,026 seconds

```
  YILLAR  SHU_YILDA_ISHGA_OLINGANLAR_SONI
1 2011                                 35
2 2012                                 35
3 2013                                 28
```

Oy bo`yicha ishga olinganlar sonini topish:



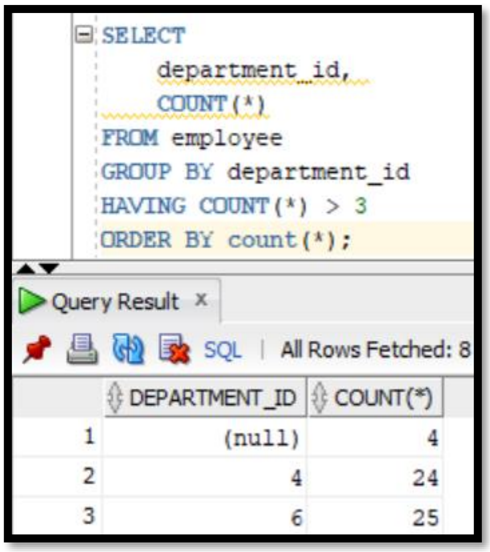
```
SELECT
    TO_CHAR(HIRE_DATE, 'YYYY') YILLAR,
    TO_CHAR(HIRE_DATE, 'MM') OYLAR,
    COUNT(*) SHU_OYDA_ISHGA_OLINGANLAR_SONI
FROM employee
GROUP BY TO_CHAR(HIRE_DATE, 'YYYY'), TO_CHAR(HIRE_DATE, 'MM')
ORDER BY COUNT(*) DESC;
```

Fetched 50 rows in 0,008 seconds

```
  YILLAR  OYLAR  SHU_OYDA_ISHGA_OLINGANLAR_SONI
1 2016    12                                  7
2 2012    01                                  6
3 2012    06                                  6
```

HAVING bandi

3tadan ko`p hodimi bo`lgan Department_id ga mansub hodimlar sonini chiqarish:



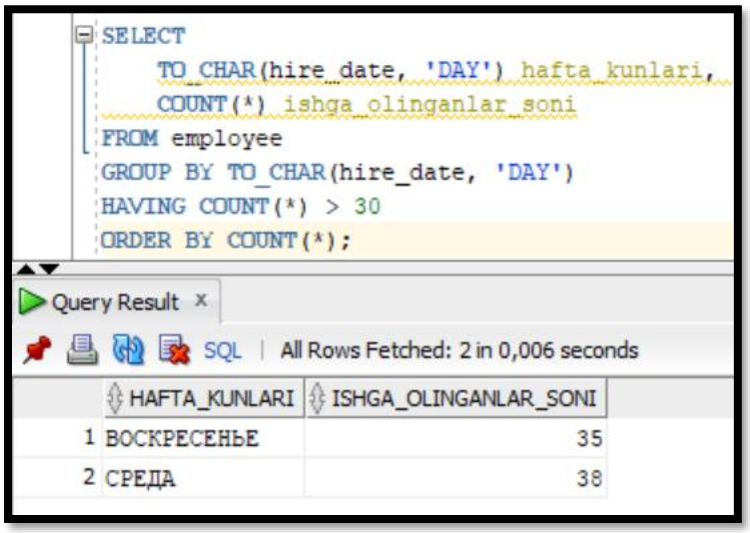
```
SELECT
    department_id,
    COUNT(*)
FROM employee
GROUP BY department_id
HAVING COUNT(*) > 3
ORDER BY count(*);
```

Query Result

All Rows Fetched: 8

```
   DEPARTMENT_ID  COUNT(*)
1         (null)         4
2              4        24
3              6        25
```

Soni 30 dan oshiq bo`lgan hafta kunlari bo`yicha hodimlar sonini chiqarish:



```
SELECT
    TO_CHAR(hire_date, 'DAY') hafta_kunlari,
    COUNT(*) ishga_olinganlar_soni
FROM employee
GROUP BY TO_CHAR(hire_date, 'DAY')
HAVING COUNT(*) > 30
ORDER BY COUNT(*);
```

Query Result

All Rows Fetched: 2 in 0,006 seconds

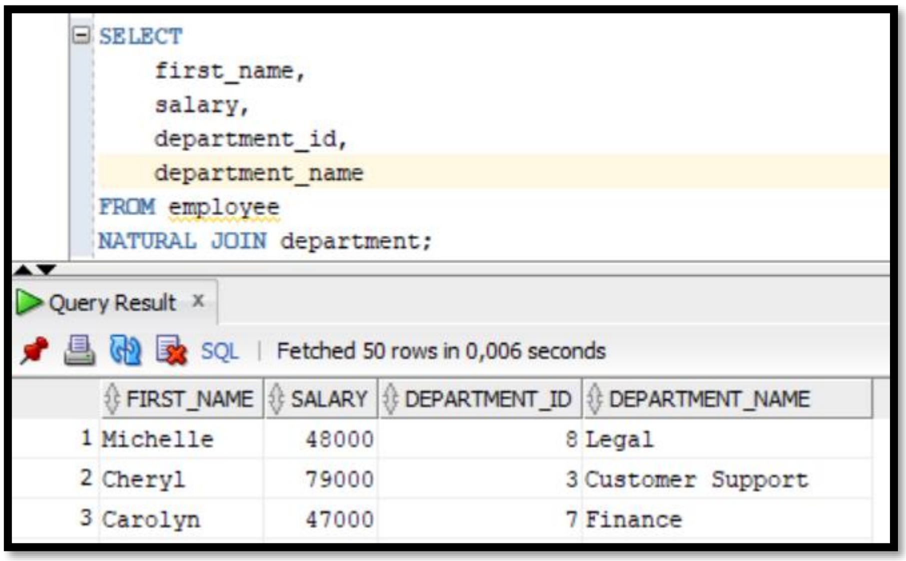
```
  HAFTA_KUNLARI  ISHGA_OLINGANLAR_SONI
1 ВОСКРЕСЕНЬЕ                       35
2 СРЕДА                             38
```

JOIN

Natural JOIN

2ta jadvalda bir xil nomli ustun bo`lsa ishlatiladi. Lekin tavsiya etilmaydi. Sababi 2ta jadvalda 2 va undan ortiq bir xil nomli ustun bo`lishi mumkin. Bunday holda aniqlik shart. Misol uchun:

employee va customer jadvallarining 2sida ham first_name va last_name bor.



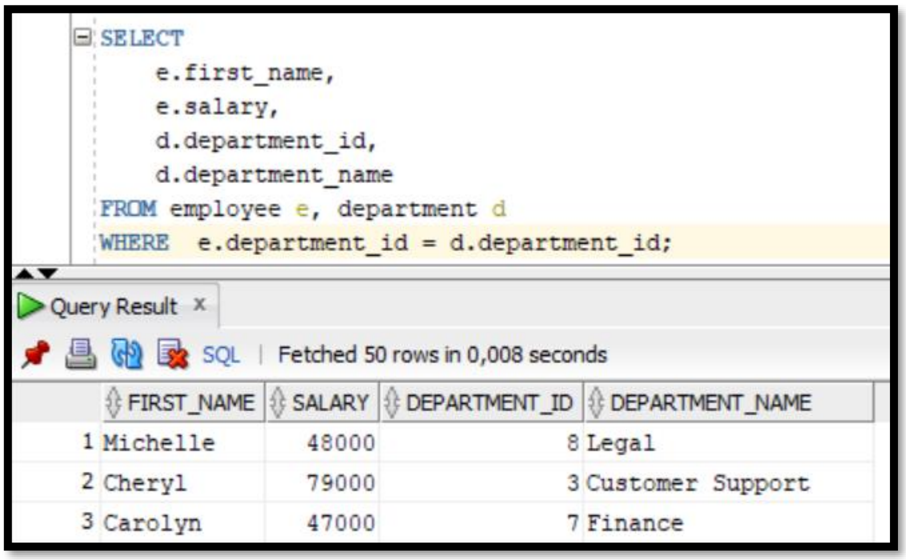
```
SELECT
    first_name,
    salary,
    department_id,
    department_name
FROM employee
NATURAL JOIN department;
```

Query Result

SQL | Fetched 50 rows in 0,006 seconds

	FIRST_NAME	SALARY	DEPARTMENT_ID	DEPARTMENT_NAME
1	Michelle	48000	8	Legal
2	Cheryl	79000	3	Customer Support
3	Carolyn	47000	7	Finance

Bunga sinonim:



```
SELECT
    e.first_name,
    e.salary,
    d.department_id,
    d.department_name
FROM employee e, department d
WHERE  e.department_id = d.department_id;
```

Query Result

SQL | Fetched 50 rows in 0,008 seconds

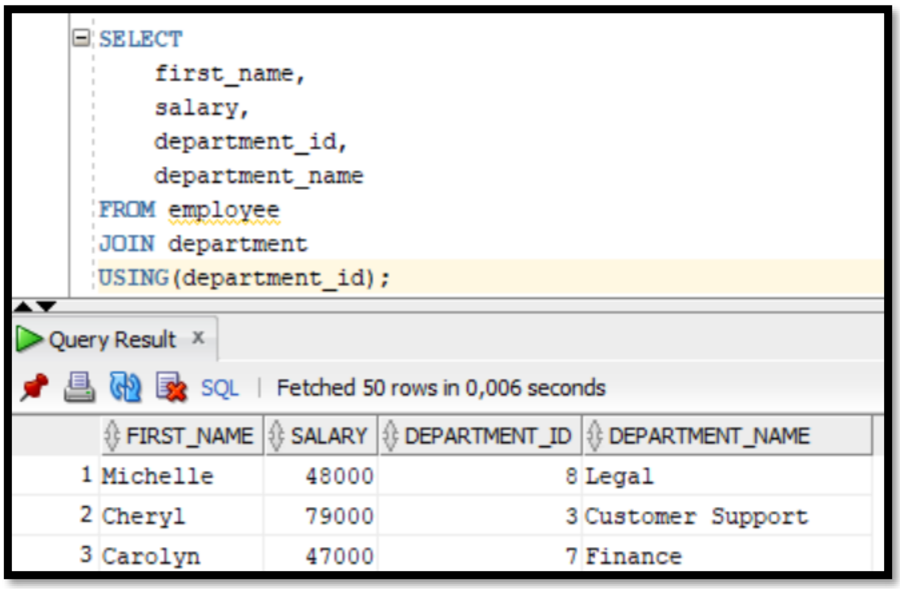
	FIRST_NAME	SALARY	DEPARTMENT_ID	DEPARTMENT_NAME
1	Michelle	48000	8	Legal
2	Cheryl	79000	3	Customer Support
3	Carolyn	47000	7	Finance

JOIN

USING ⇔ WHERE e.department_id = d.department_id;

USING o`xshash ustunlar nomini kiritish uchun ishlatiladi.

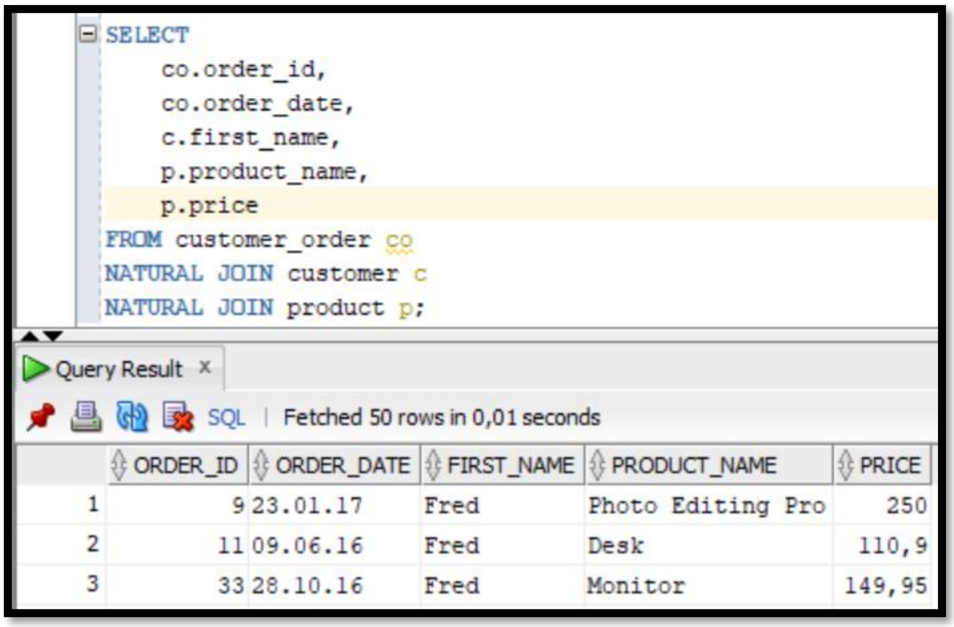
USING ishlatilganda jadvallarga taxallus qo`yilsa xatolik chiqaradi.



```
SELECT
    first_name,
    salary,
    department_id,
    department_name
FROM employee
JOIN department
USING(department_id);
```

	FIRST_NAME	SALARY	DEPARTMENT_ID	DEPARTMENT_NAME
1	Michelle	48000	8	Legal
2	Cheryl	79000	3	Customer Support
3	Carolyn	47000	7	Finance

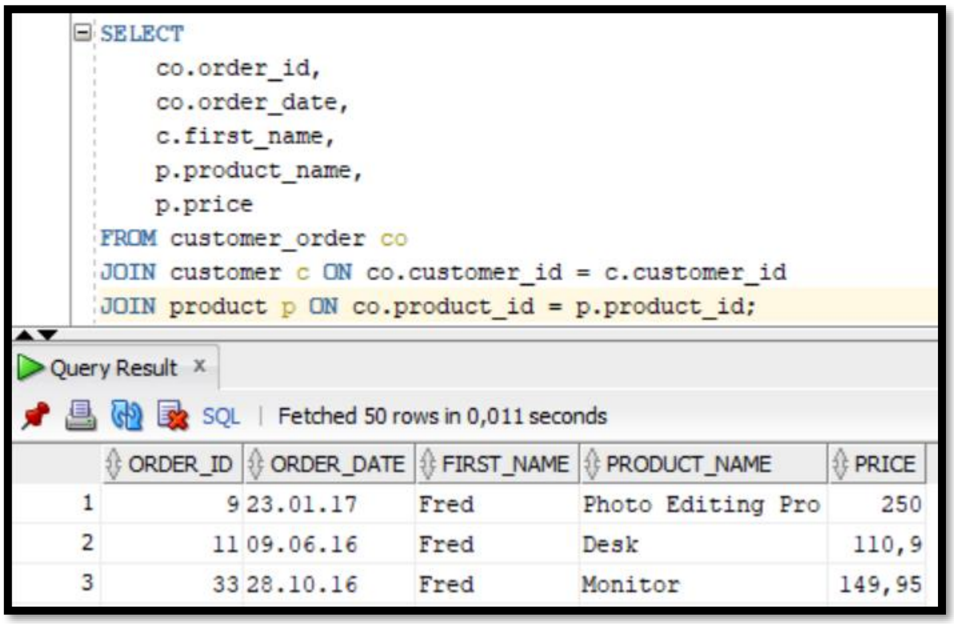
JOINing Multiple Tables



```
SELECT
    co.order_id,
    co.order_date,
    c.first_name,
    p.product_name,
    p.price
FROM customer_order co
NATURAL JOIN customer c
NATURAL JOIN product p;
```

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

Bunga sinonim:

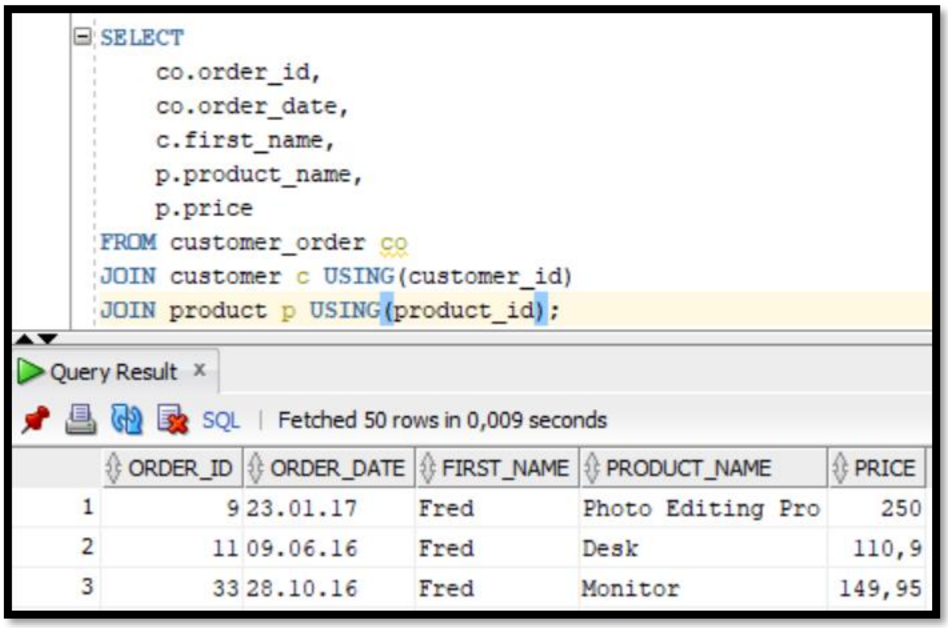

```
SELECT
    co.order_id,
    co.order_date,
    c.first_name,
    p.product_name,
    p.price
FROM customer_order co
JOIN customer c ON co.customer_id = c.customer_id
JOIN product p ON co.product_id = p.product_id;
```

Query Result ×

SQL | Fetched 50 rows in 0,011 seconds

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

Yana bitta sinonim:


```
SELECT
    co.order_id,
    co.order_date,
    c.first_name,
    p.product_name,
    p.price
FROM customer_order co
JOIN customer c USING(customer_id)
JOIN product p USING(product_id);
```

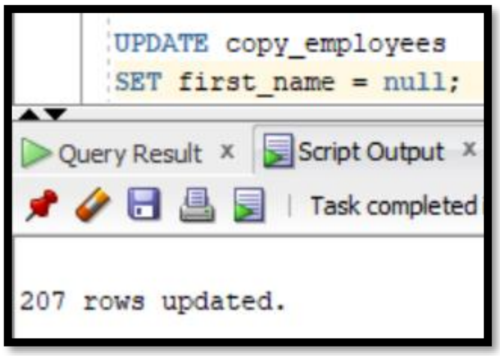
Query Result ×

SQL | Fetched 50 rows in 0,009 seconds

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

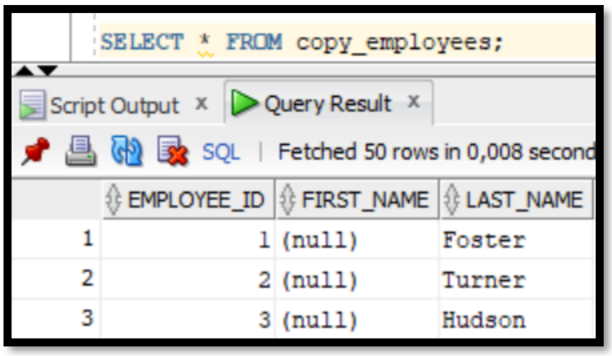
Insert, Update, Delete

DML Sentence



```
UPDATE copy_employees
SET first_name = null;
```

```
207 rows updated.
```

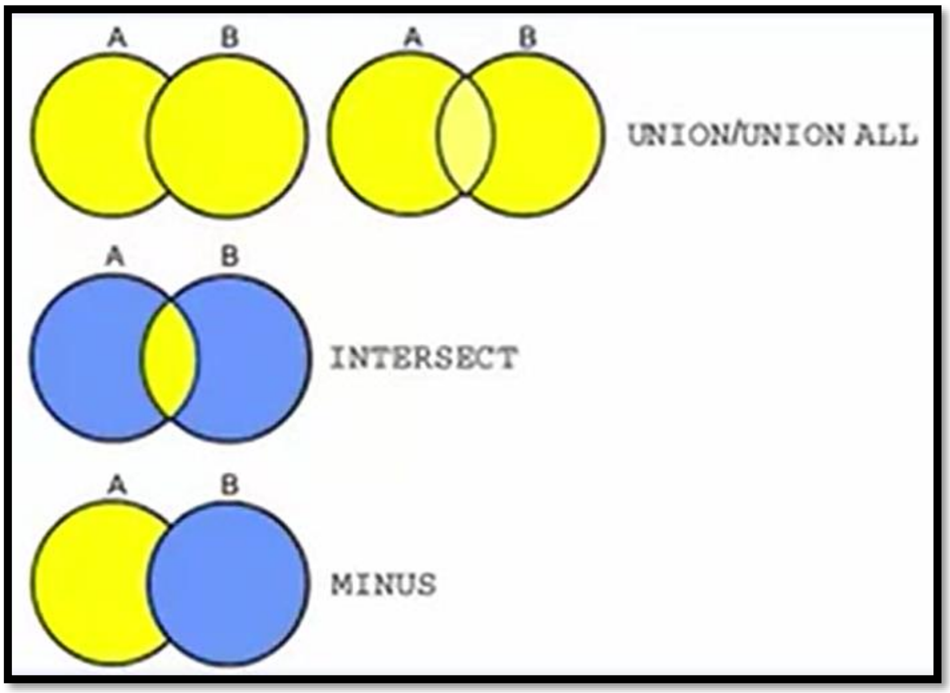


```
SELECT * FROM copy_employees;
```

Fetched 50 rows in 0,008 second

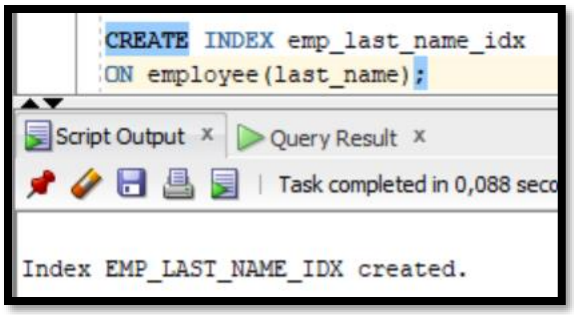
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	1	(null)	Foster
2	2	(null)	Turner
3	3	(null)	Hudson

Operator



Index

Indeks qatorlarni qidirishni tezlashtiradi.



```
CREATE INDEX emp_last_name_idx
ON employee(last_name);
```

```
Index EMP_LAST_NAME_IDX created.
```

Sequence

Sequence ⇔ Auto_increment

*Sequence*lar jadvallardan ajratilgan holda ishlaydi.

Bir nechta jadval uchun bitta ketma-ketlikdan foydalanishimiz mumkin.

```
CREATE SEQUENCE sequence_name
  INCREMENT BY interval
  START WITH first_number
  MINVALUE min_value | NOMINVALUE
  MAXVALUE max_value | NOMAXVALUE
  CYCLE | NOCYCLE
  CAChE cache_value | NOCAChE
  ORDER | NOORDER;
```

Misol: 50 dan boshlanib, 25ga ortib boruvchi ketma-ketlik. Maksimal qiymati 100. CYCLE bo`lgani uchun 100 dan keyingi qiymat yana boshidagi qiymatga ya'ni 50ga qaytadi.

```
CREATE SEQUENCE id_seq
  INCREMENT BY 25
  START WITH 50
  MINVALUE 50
  MAXVALUE 100
  CYCLE
  CAChE 2;
```

Ketma-ketlikning keyingi qiymatini olish:

```
SELECT id_seq.NEXTVAL
FROM dual;
```

Ketma-ketlikning joriy qiymatini olish:

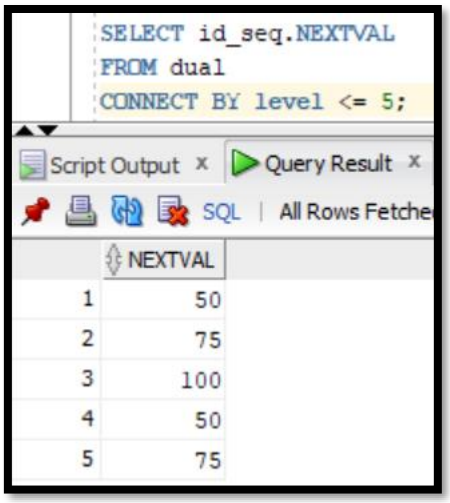
```
SELECT id_seq.CURRVAL
FROM dual;
```

Drop sequence:

```
DROP SEQUENCE sequence_name;
```

Ushbu SELECT bayonot id_seq.NEXTVAL qiymatni qayta-qayta ishlatadi:

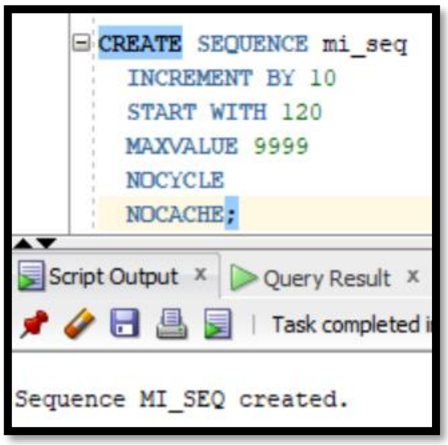
```
SELECT id_seq.NEXTVAL
FROM dual
CONNECT BY level <= 5;
```



```
SELECT id_seq.NEXTVAL
FROM dual
CONNECT BY level <= 5;
```

```
   NEXTVAL
1       50
2       75
3      100
4       50
5       75
```

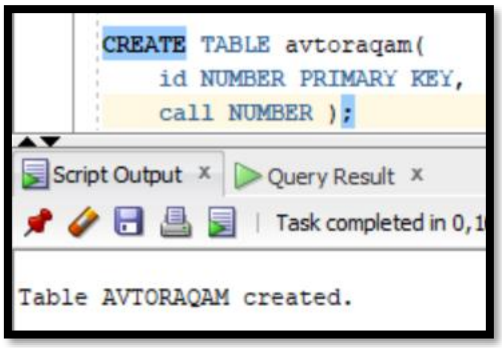
Sequence yaratildi:



```
CREATE SEQUENCE mi_seq
  INCREMENT BY 10
  START WITH 120
  MAXVALUE 9999
  NOCYCLE
  NOCACHE;
```

```
Sequence MI_SEQ created.
```

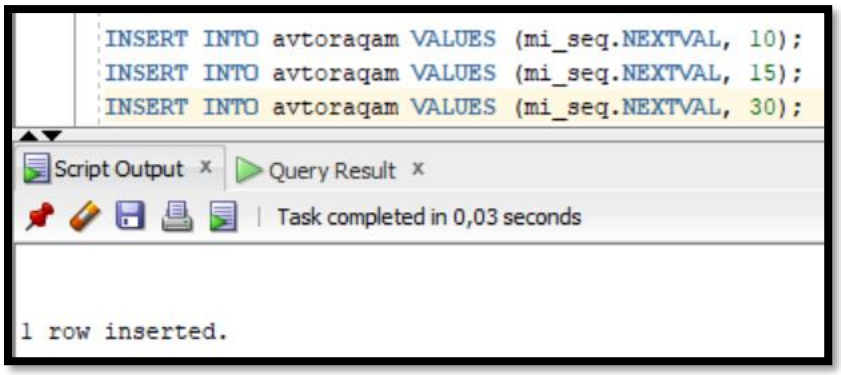
Jadval yaratildi:



```
CREATE TABLE avtoraqam(
    id NUMBER PRIMARY KEY,
    call NUMBER );
```

```
Table AVTORAQAM created.
```

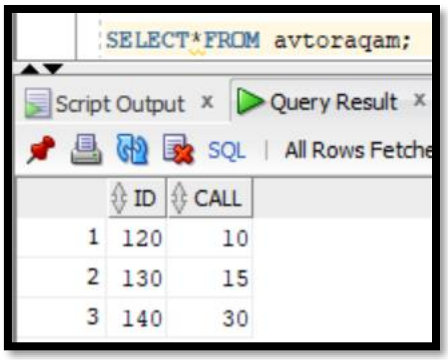
Jadvalga ma’lumotlar kiritildi:



```
INSERT INTO avtoraqam VALUES (mi_seq.NEXTVAL, 10);
INSERT INTO avtoraqam VALUES (mi_seq.NEXTVAL, 15);
INSERT INTO avtoraqam VALUES (mi_seq.NEXTVAL, 30);
```

```
1 row inserted.
```

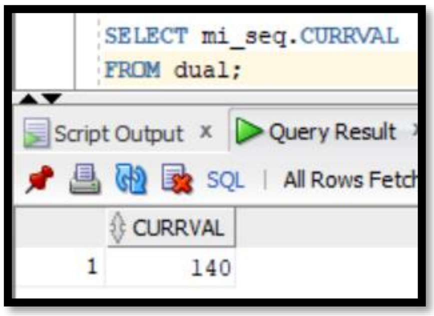
Sequenceni tekshiramiz:



```
SELECT*FROM avtoraqam;
```

	ID	CALL
1	120	10
2	130	15
3	140	30

Hozirgi qiymatni aniqlaymiz:

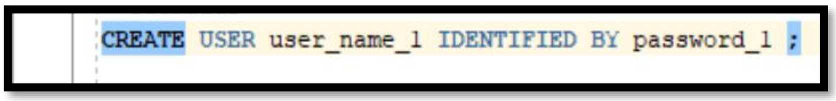


```
SELECT mi_seq.CURRVAL
FROM dual;
```

```
   CURRVAL
1      140
```

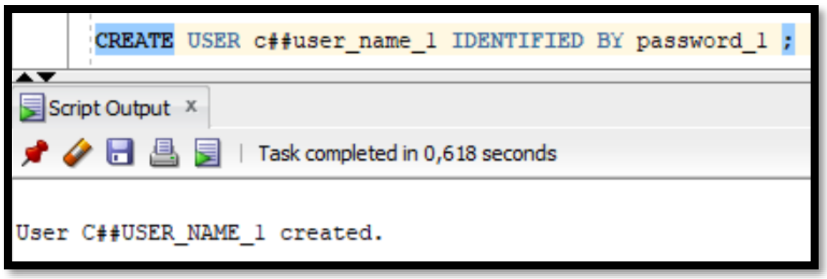
Sxemalar

User yaratish:



```
CREATE USER user_name_1 IDENTIFIED BY password_1 ;
```

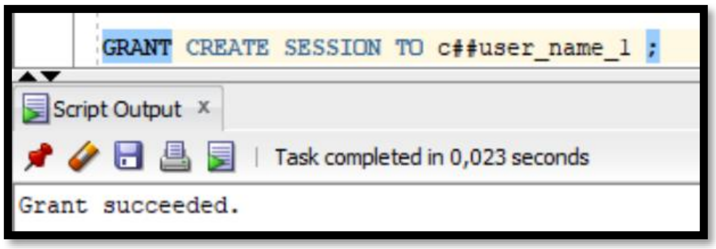
Nomini rad etish holati kuzatilsa:



```
CREATE USER c##user_name_1 IDENTIFIED BY password_1 ;
```

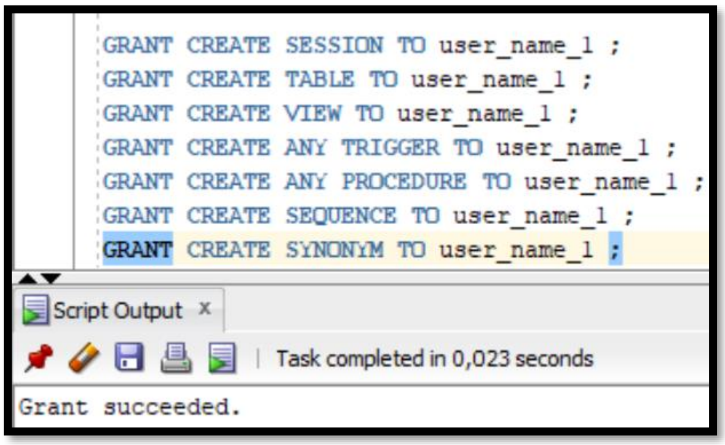
```
Task completed in 0,618 seconds

User C##USER_NAME_1 created.
```

USER larga imtiyoz berish: GRANT



```
GRANT CREATE SESSION TO c##user_name_1 ;
```

```
Task completed in 0,023 seconds
Grant succeeded.
```

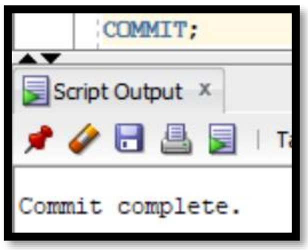


```
GRANT CREATE SESSION TO user_name_1 ;
GRANT CREATE TABLE TO user_name_1 ;
GRANT CREATE VIEW TO user_name_1 ;
GRANT CREATE ANY TRIGGER TO user_name_1 ;
GRANT CREATE ANY PROCEDURE TO user_name_1 ;
GRANT CREATE SEQUENCE TO user_name_1 ;
GRANT CREATE SYNONYM TO user_name_1 ;
```

```
Task completed in 0,023 seconds
Grant succeeded.
```

Transaction

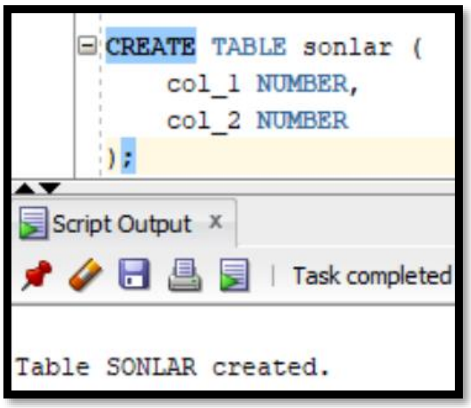
Tranzaksiyada topshiriq boshlanishi/tugashini anglatadi: *COMMIT*



```
COMMIT;
```

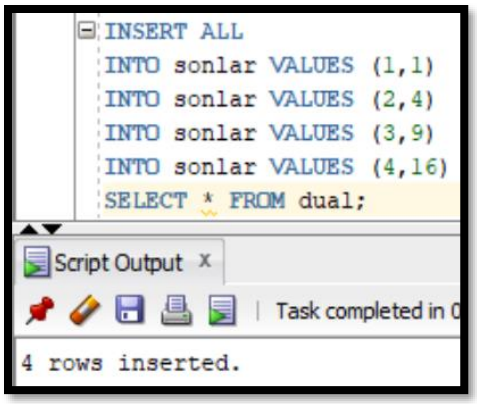
```
Commit complete.
```

Jadval yaratib, qiymatlar kiritamiz va tekshiramiz:



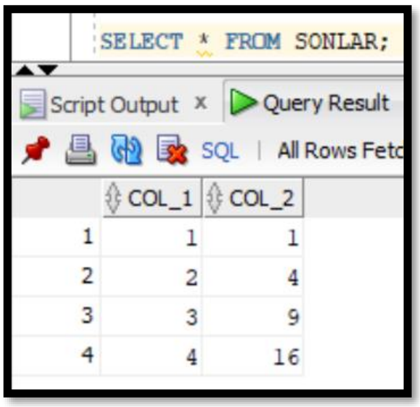
```
CREATE TABLE sonlar (
    col_1 NUMBER,
    col_2 NUMBER
);
```

```
Table SONLAR created.
```



```
INSERT ALL
INTO sonlar VALUES (1,1)
INTO sonlar VALUES (2,4)
INTO sonlar VALUES (3,9)
INTO sonlar VALUES (4,16)
SELECT * FROM dual;
```

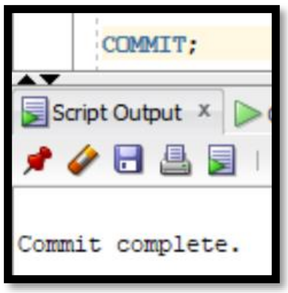
```
4 rows inserted.
```



```
SELECT * FROM SONLAR;
```

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16

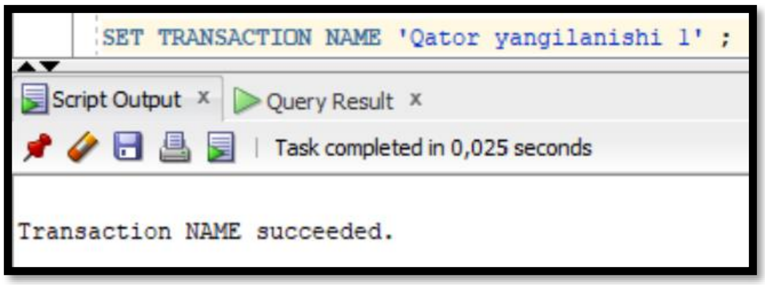
Tranzaksiya topshiriqlarini tugatish:



```
COMMIT;
```

```
Commit complete.
```

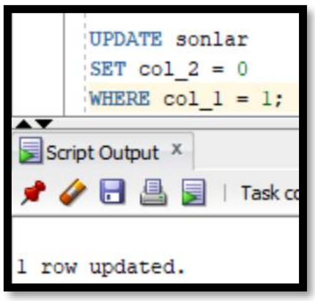
Tranzaksiya o`rnatish: *SET TRANSACTION*



```
SET TRANSACTION NAME 'Qator yangilanishi 1' ;
```

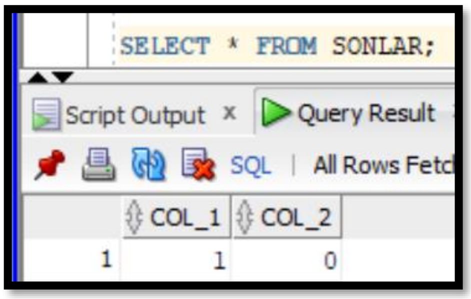
```
Transaction NAME succeeded.
```

Yangilanish kiritib, tekshiramiz va *SAVEPOINT* ni belgilaymiz:



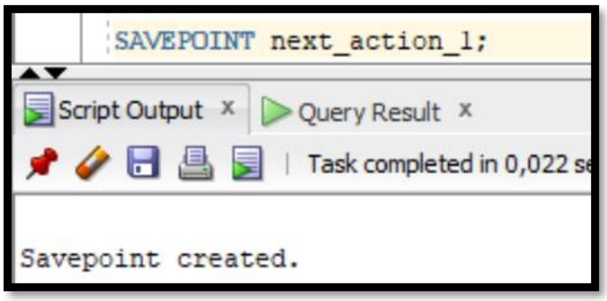
```
UPDATE sonlar
SET col_2 = 0
WHERE col_1 = 1;
```

```
1 row updated.
```



```
SELECT * FROM SONLAR;
```

	COL_1	COL_2
1	1	0

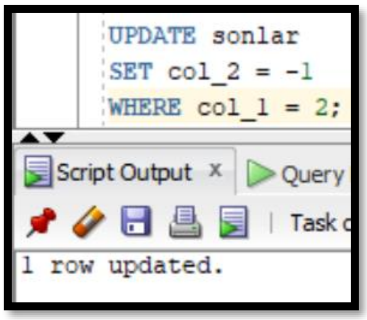


```
SAVEPOINT next_action_1;
```

```
Task completed in 0,022 se
```

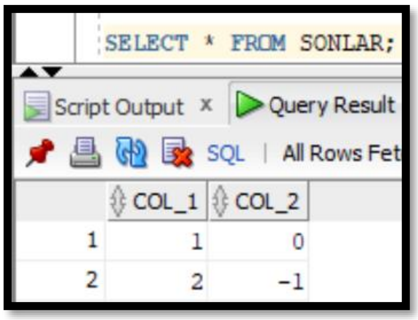
```
Savepoint created.
```

Yangilanish kiritib, tekshiramiz va SAVEPOINT 2-sini belgilaymiz:



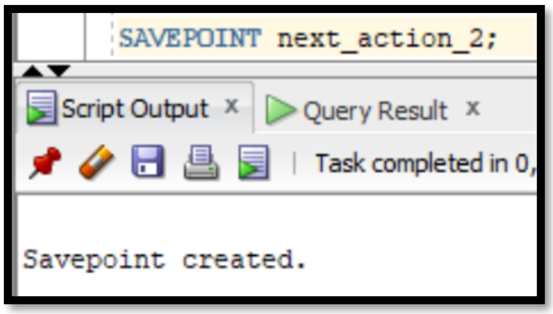
```
UPDATE sonlar
SET col_2 = -1
WHERE col_1 = 2;
```

```
1 row updated.
```



```
SELECT * FROM SONLAR;
```

	COL_1	COL_2
1	1	0
2	2	-1

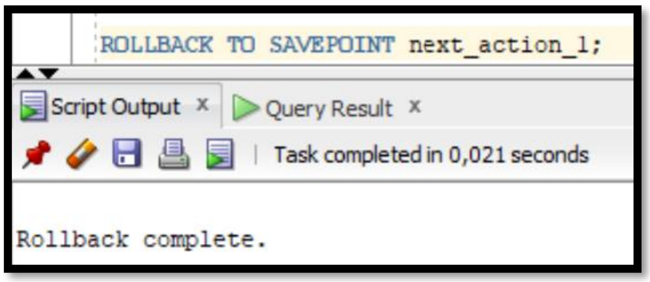


```
SAVEPOINT next_action_2;
```

```
Task completed in 0,
```

```
Savepoint created.
```

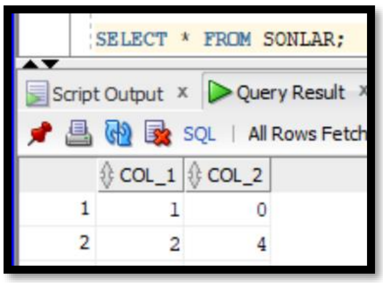
1-SAVEPOINTga qaytamiz va tekshiramiz:



```
ROLLBACK TO SAVEPOINT next_action_1;
```

```
Task completed in 0,021 seconds
```

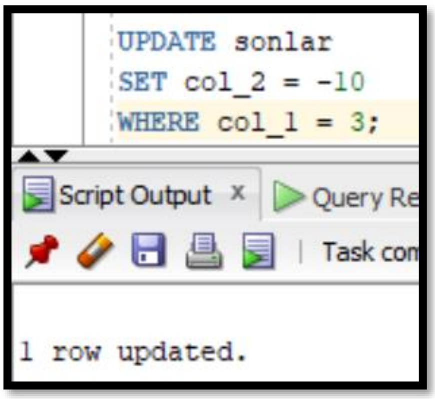
```
Rollback complete.
```



```
SELECT * FROM SONLAR;
```

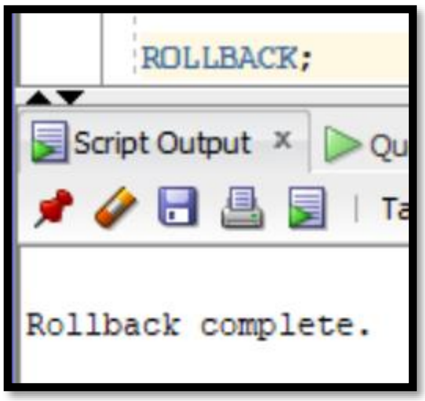
	COL_1	COL_2
1	1	0
2	2	4

Yangilanish kiritib, boshlang`ich holatga qaytaramiz va tekshiramiz:



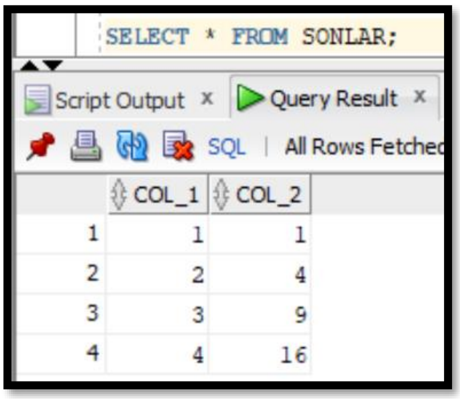
```
UPDATE sonlar
SET col_2 = -10
WHERE col_1 = 3;
```

```
1 row updated.
```



```
ROLLBACK;
```

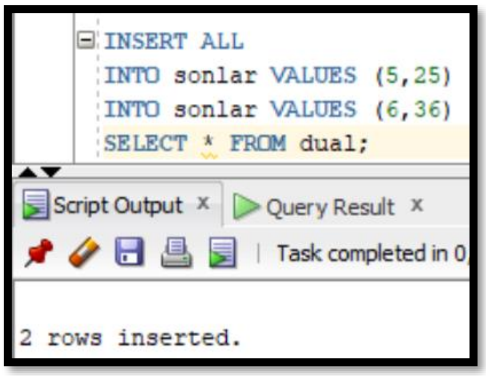
```
Rollback complete.
```



```
SELECT * FROM SONLAR;
```

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16

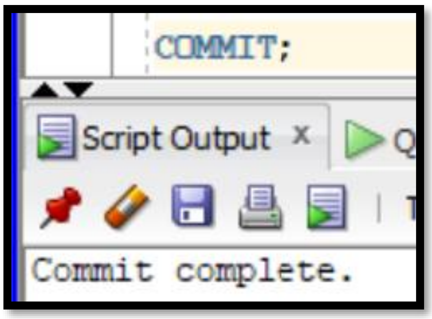
Yangilanish kiritamiz:



```
INSERT ALL
INTO sonlar VALUES (5,25)
INTO sonlar VALUES (6,36)
SELECT * FROM dual;
```

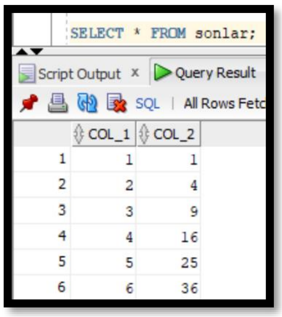
```
2 rows inserted.
```

COMMIT qilib, tekshiramiz va DELETE qilamiz:



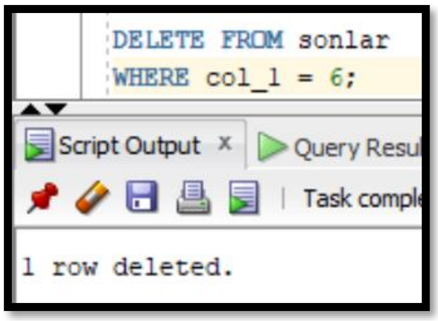
```
COMMIT;
```

```
Commit complete.
```



```
SELECT * FROM sonlar;
```

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25
6	6	36



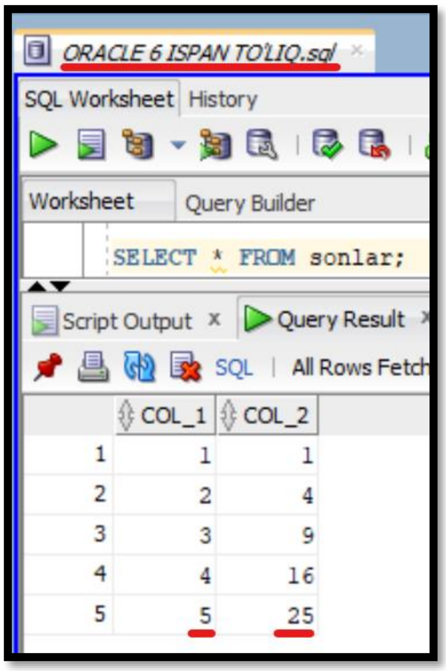
```
DELETE FROM sonlar
WHERE col_1 = 6;
```

```
1 row deleted.
```

Ispan bazasini va Turk bazasini tekshiramiz:

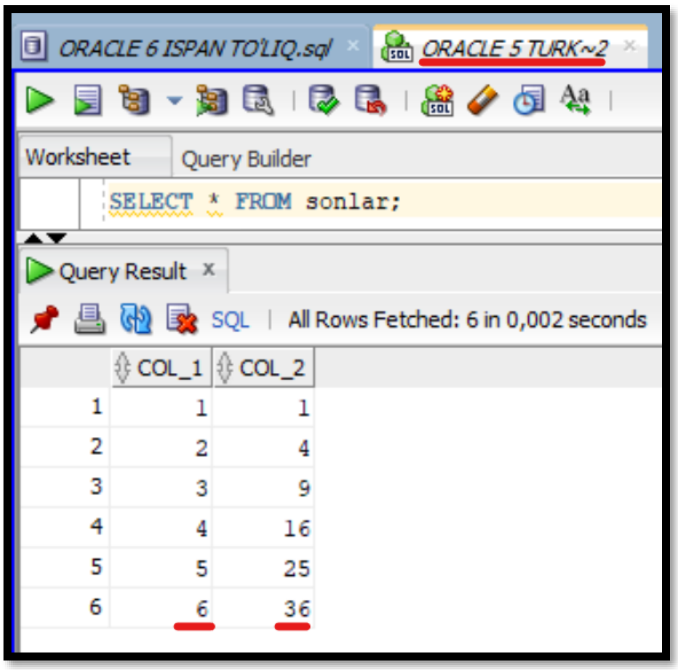
Natija. Ispanda bajarilgan DELETE Turk bazasida amalga oshirilmagan.

Sabab. Tranzaksiya hali to`liq yakunlanmadi.



```
SELECT * FROM sonlar;
```

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25



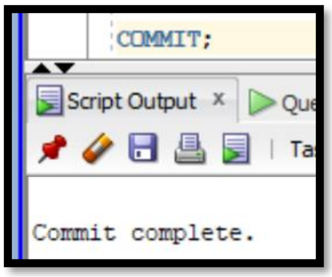
ORACLE 6 ISPAN TO'LIQ.sql
ORACLE 5 TURK~2
Worksheet
Query Builder

```
SELECT * FROM sonlar;
```

Query Result
SQL
All Rows Fetched: 6 in 0,002 seconds

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25
6	6	36

To`liq yakunlash uchun COMMIT qilamiz:

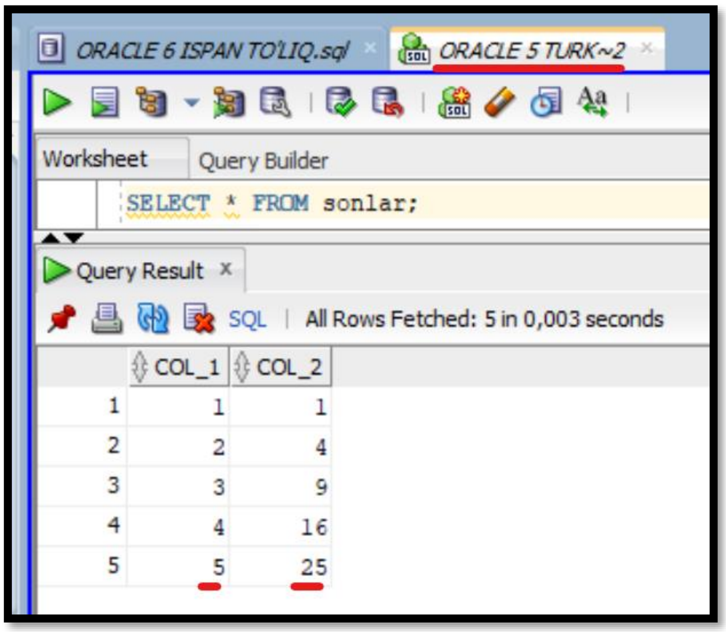


```
COMMIT;
```

```
Commit complete.
```

Endi esa Turk bazasida tekshiramiz:

Natija DELETE vazifasi bajarilgan.



```
SELECT * FROM sonlar;
```

All Rows Fetched: 5 in 0,003 seconds

	COL_1	COL_2
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

VIEW

View bu virtual jadval bo`lib, nomlangan so`rovdir. Asosiy jadval ustuni nomi o`zgartirilsa yoki o`chirilsa u ishlamay qoladi.

Ko`rinishlardan quyidagi maqsadlarda foydalanishingiz mumkin:

- Ma’lumotlarni qidirishni soddalashtirish. (murakkab so`rovni qayta-qayta yozmaysiz)
- Mantiqiy ma’lumotlarning mustaqilligini saqlash.
- Ma’lumotlar xavfsizligini ta’minlash.

View yaratish:

```
CREATE VIEW view_name AS
  SELECT columns
  FROM tables
  [WHERE conditions];
```

Uni chaqirish:

```
SELECT * FROM view_name;
```

VIEW ni yangilang

```
CREATE OR REPLACE VIEW view_name AS
  SELECT columns
  FROM table;
```

Viewni o`chirish

```
DROP VIEW view_name;
```

Index

Indeks - bu yozuvlarni tezroq olish imkonini beruvchi unumdorlikni sozlash usuli. Indeks indekslangan ustunlarda paydo bo`ladigan har bir qiymat uchun yozuv yaratadi. Odatda Oracle B-tree indekslarini yaratadi.

B-tree index

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (column1, column2, ... column_n)
  [ COMPUTE STATISTICS ];
```

Misol:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name);
```

Funktsiyaga asoslangan indeks yaratish

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (function1, function2, ... function_n)
  [ COMPUTE STATISTICS ];
```

Misol:

```
CREATE INDEX supplier_idx
  ON supplier (UPPER(supplier_name));
```

Indeks nomini o`zgartirish

```
ALTER INDEX index_name
  RENAME TO new_index_name;
```

Misol:

```
ALTER INDEX supplier_idx
  RENAME TO supplier_index_name;
```

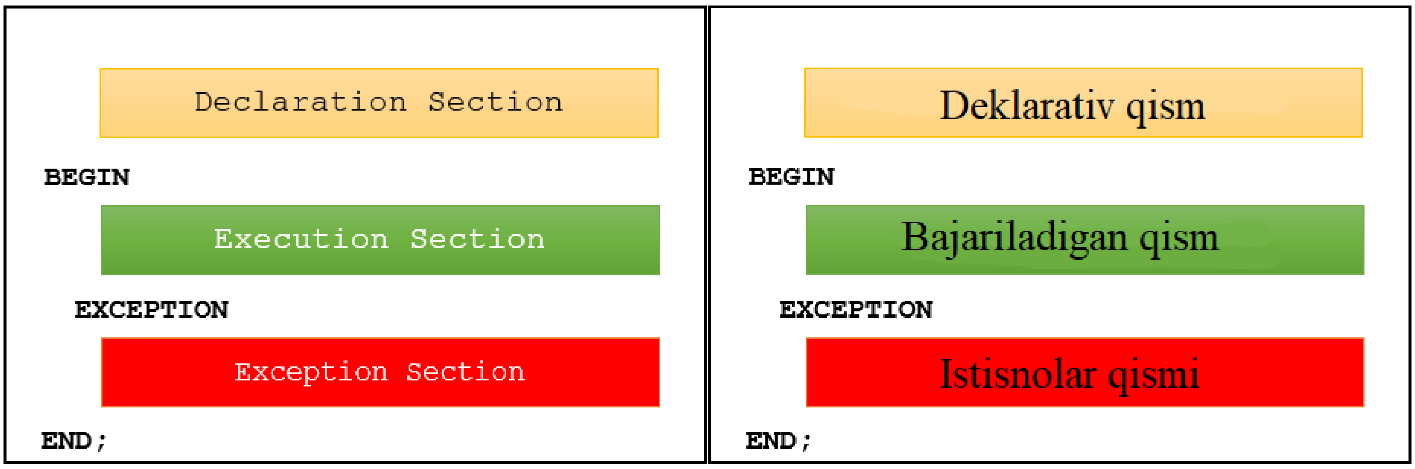
Indeksni o`chirish

```
DROP INDEX index_name;
```

Procedure

Protsedura va funksiya – bu ma’lum bir vazifani bajaradigan dastur bloki.

PL/SQL blokining tuzilishi:



```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (parameter [,parameter]) ]
IS
    [Declaration_section]
BEGIN
    [Execution_section]
    [Exception_section]
END [procedure_name];
```

1) Declaration section ⇔ Deklaratsiya bo`limi

PL/SQL blokida siz o`zgaruvchilarni e'lon qiladigan , kursorlar uchun xotira ajratadigan va ma'lumotlar turlarini aniqlaydigan deklaratsiya bo`limi mavjud. Bu qismda siz variables (o`zgaruvchilar), constants (doimiylar), cursors va hokazolarni e'lon qilishingiz mumkin.

2) Execution section ⇔ Bajariladigan bo`lim

Bajariladigan bo`lim **BEGIN** kalit so`zi bilan boshlanadi va **END** kalit so`zi bilan tugaydi. Ushbu bo`limda kamida bitta bajariladigan bayonot bo`lishi kerak, hatto u hech narsa qilmaydigan NULL bayonot bo`lsa ham.

3) Exception-handling section ⇔ Istisnolarni qayta ishlash bo`limi

PL/SQL blokida EXCEPTION kalit so`zi bilan boshlanadigan istisnolarni qayta ishlash bo`limi mavjud. Istisnolarni ko`rib chiqish bo`limi - bu ijro bo`limidagi kod tomonidan ko`tarilgan istisnolarni ushlaysiz va boshqarasiz.

Protsedura yoki funktsiyani yaratganingizda, siz parametrlarni belgilashingiz mumkin. E'lon qilinishi mumkin bo`lgan uchta turdagi parametrlar mavjud:

1. **IN** – bu parametr faqat o`qish uchun mo`ljallangan. Bu parametrga protsedura va funksiya orqali murojaat qilish mumkin, lekin uning qiymatini o`zgartira olmaysiz. Oracle IN dan standart rejim sifatida foydalanadi. Ya'ni agar siz parametr uchun rejimni aniq belgilamasangiz, Oracle IN rejimdan foydalanadi.
2. **OUT** – bu parametr faqat qiymat yozish uchun mo`ljallangan. Parametrga protsedura yoki funksiya orqali (o`qish) murojaat qilish mumkin emas.
3. **IN OUT** - Parametrga murojaat qilish va parametr qiymatini yozish mumkin. (ham o`qilishi, ham yozilishi mumkin.)

Misol:

```
CREATE OR REPLACE Procedure UpdateCourse

   ( name_in IN varchar2 )

IS

   cnumber number;


   cursor c1 is

   SELECT course_number

    FROM courses_tbl

    WHERE course_name = name_in;

BEGIN

   open c1;

   fetch c1 into cnumber;


   if c1%notfound then

      cnumber := 9999;

   end if;


   INSERT INTO student_courses

   ( course_name,

     course_number )

   VALUES

   ( name_in,

     cnumber );


   commit;

   close c1;

EXCEPTION

WHEN OTHERS THEN

   raise_application_error(-20001,’An error was encountered -
‘||SQLCODE||’ -ERROR- ‘||SQLERRM);

END;
```

Protsedurani o`chirish

```
DROP PROCEDURE procedure_name;
```

PL/SQL protsedurasi misolini yaratish

Quyidagi protsedura mijoz identifikatorini qabul qiladi va mijozning ismi, familiyasi va elektron pochtasi kabi aloqa ma’lumotlarini chop etadi:

```
CREATE [OR REPLACE] PROCEDURE print_contact(
        p_person_id NUMBER )
IS
  r_contact persons%ROWTYPE;
BEGIN
  -- get contact based on customer id
   SELECT *
   INTO r_contact
   FROM persons
   WHERE person_id = p_person_id;

   -- print out contact’s information
   dbms_output.put_line( r_contact.first_name || '' ||
   r_contact.last_name || '<' || r_contact.contact || '>' );

EXCEPTION
  WHEN OTHERS THEN
     dbms_output.put_line( SQLERRM );
END;
```

Quyida protsedurani bajarish sintaksisi ko`rsatilgan:

```
EXECUTE procedure_name( arguments);
```

Yoki

```
EXEC procedure_name( arguments);
```

Masalan, mijoz identifikatori 100 kontakt ma’lumotlarini chop etish print_contact protsedurasini bajarish uchun siz quyidagi bayonotdan foydalanasiz:

```
EXEC print_contact(100);
```