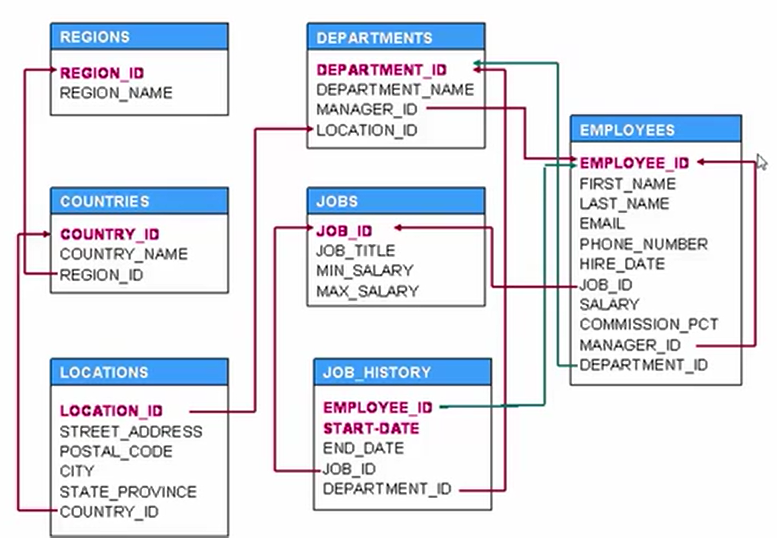


# 1-qadam

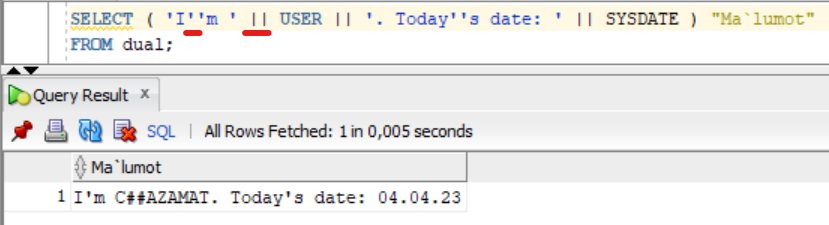
## Ma’lumotlar modeli



Text chiqarish:

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

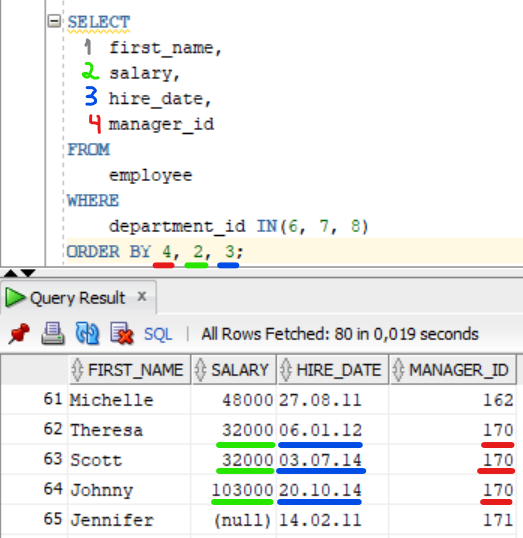
{ **||** } bu belgi ketma-ketliklarni qo`shib chiqar ma’nosini anglatadi.



## ORDER BY bandi

Ustunlarni tartib nomeri bo`yicha saralash:

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



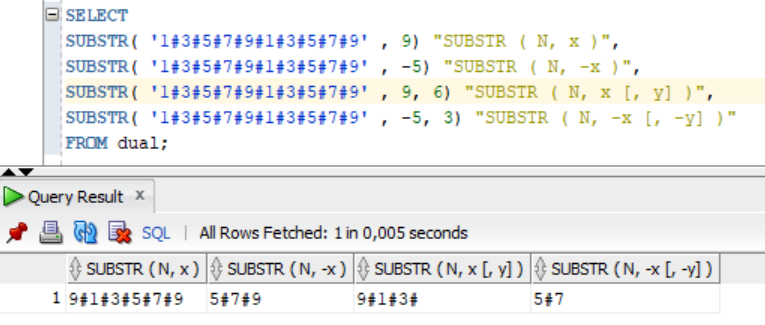
## Substr funksiyasi

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.

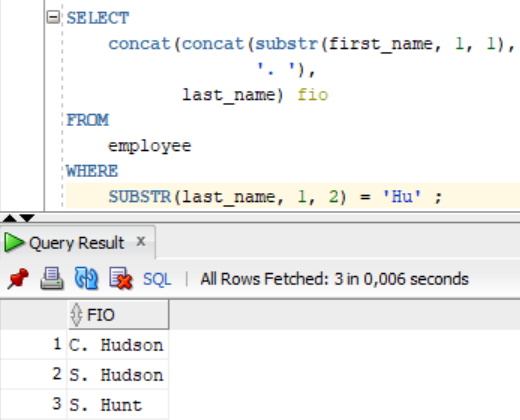
SUBSTR ( N, x [, y] ) 🡪 x-belgidan boshlab, y ta belgi yozadi.

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



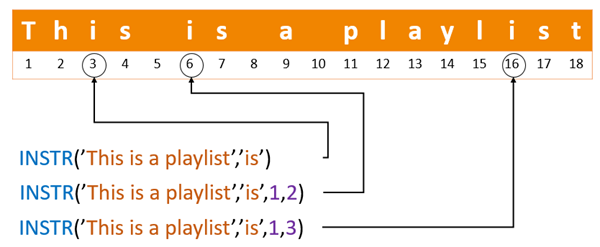
Ism familiyani qisqartirish:

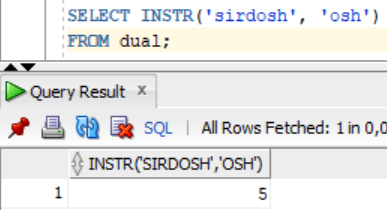
*( SUBSTR*ni *WHERE* bandida ham ishlatsa bo`ladi *)*

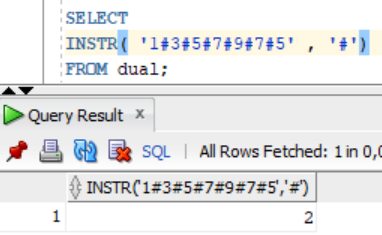
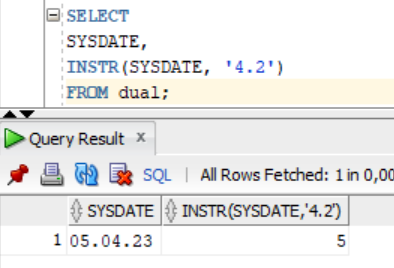


## INSTR()

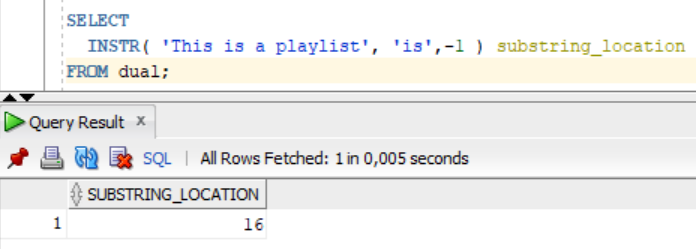
INSTR ⬄ berilgan matndan belgilangan belgilarni tartib nomerini ko`rsatadi.



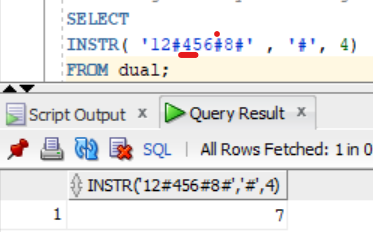




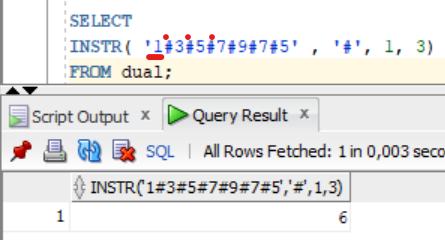
Oxiridan boshiga qarab birinchi takrorlanishni qidirish:



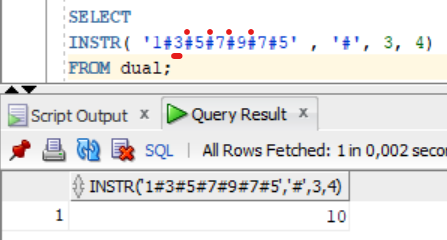
🡻 4- belgidan keyin keladigan # ni tartib nomerini topish:



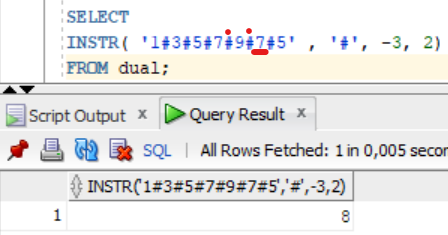
1-belgidan keyin keladigan 3- marta takrorlanadigan # ni tartib nomerini topish:



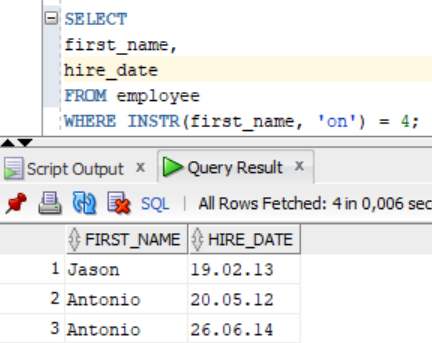
3-belgidan keyin keladigan 4-marta takrorlanadigan # ni tartib nomerini topish:



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

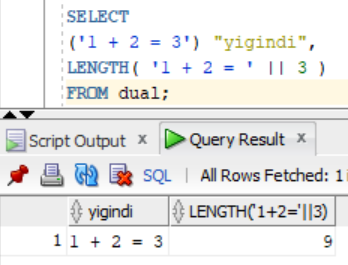
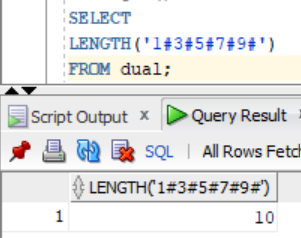


*INSTR* ni *WHERE* bandida ishlatilishi:



## LENGTH() funksiyasi

Belgilar sonini aniqlash:



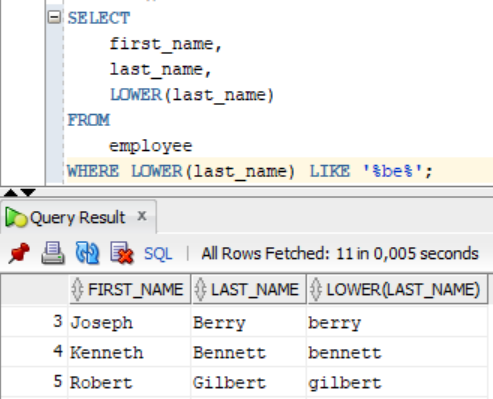
Where bandida ishlatilishi:



## LOWER() Funksiyasi

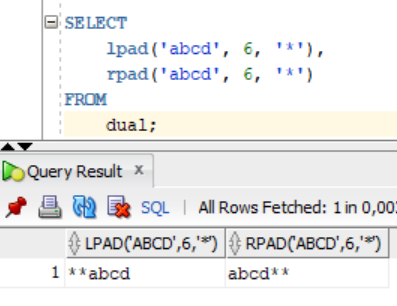
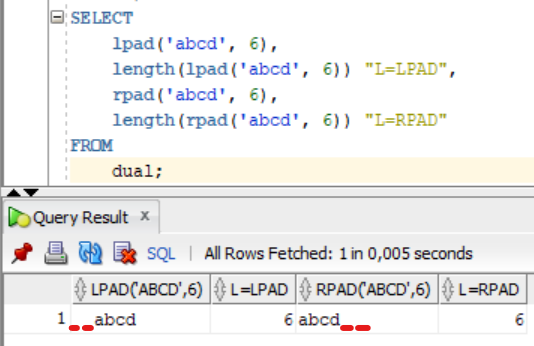
Lower 🡪 kichik harflarga o`tkazadi.

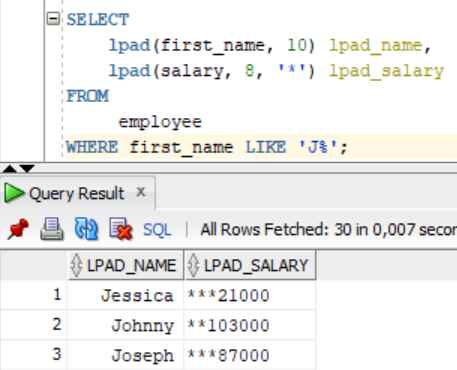
Where bandida ishlatilishi:



## LPAD, RPAD

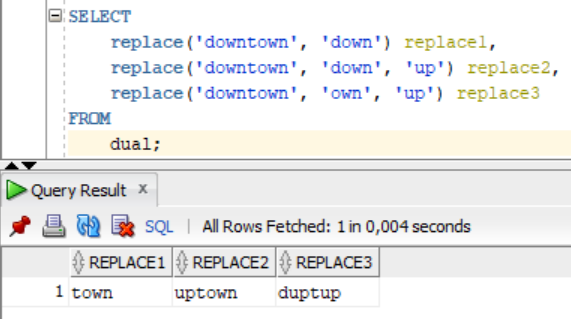
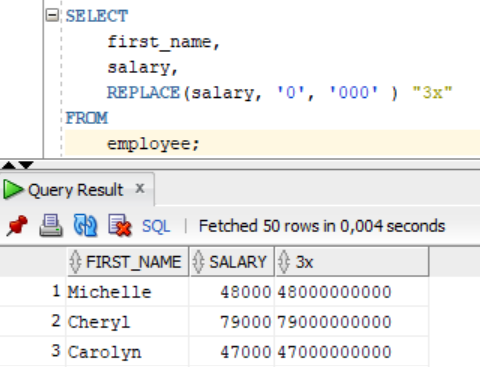
Lpad(c1, x1 [, y1])



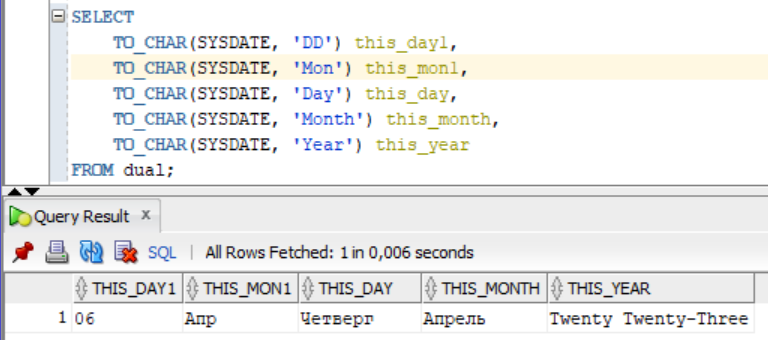


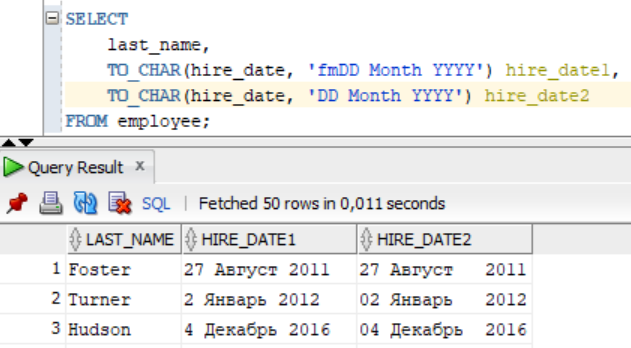
## REPLACE

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

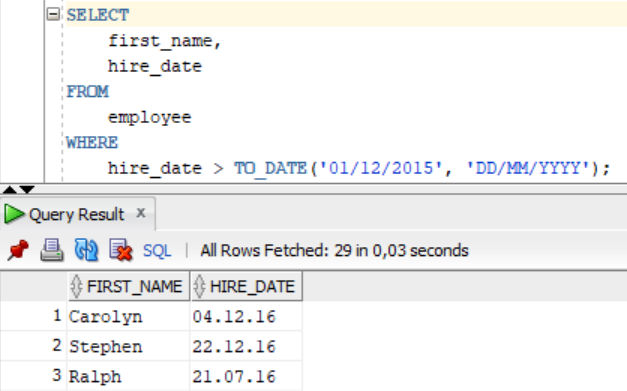
 

## TO\_ChAR

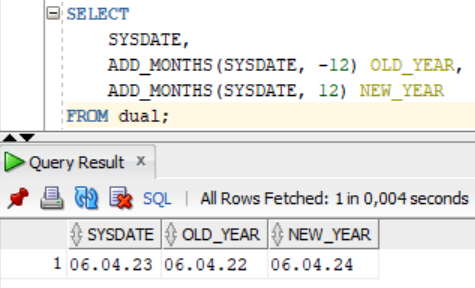




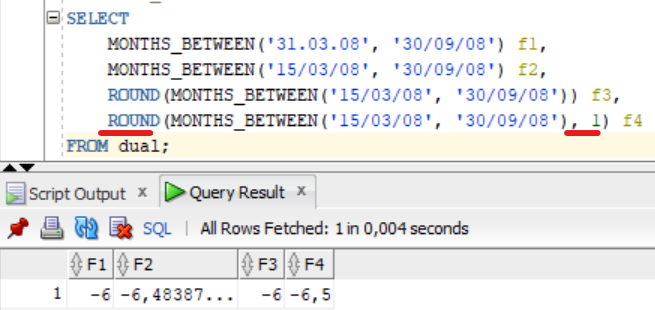
## TO\_DATE



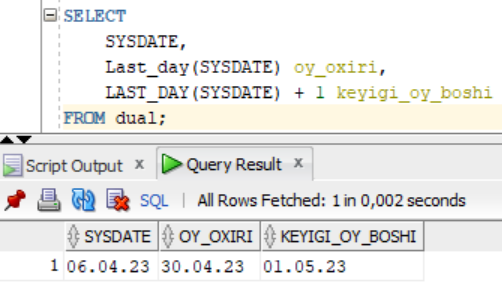
## ADD\_MONTHS



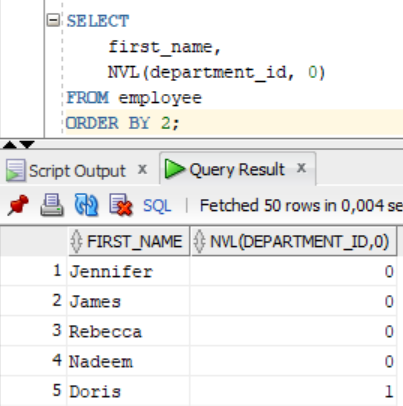
## MONTHS\_BETWEEN

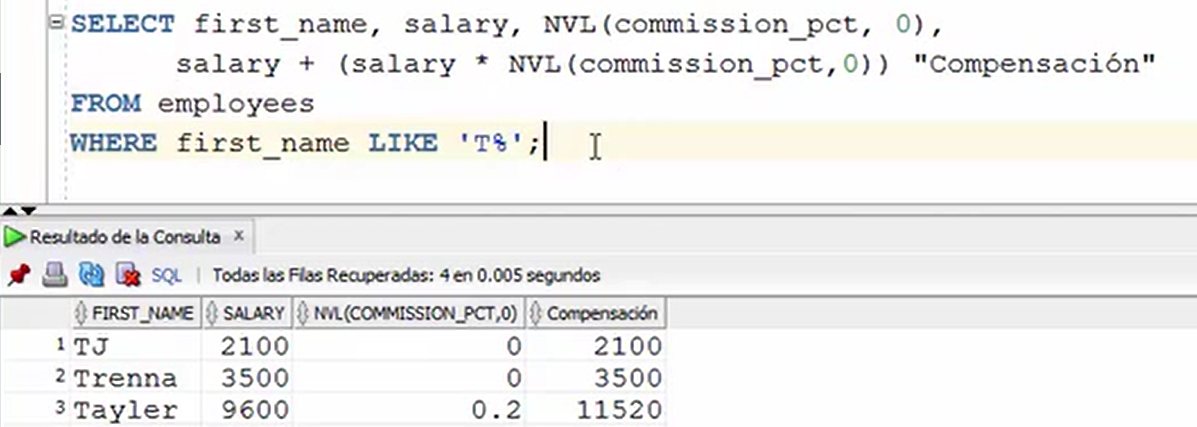


## LAST\_DAY

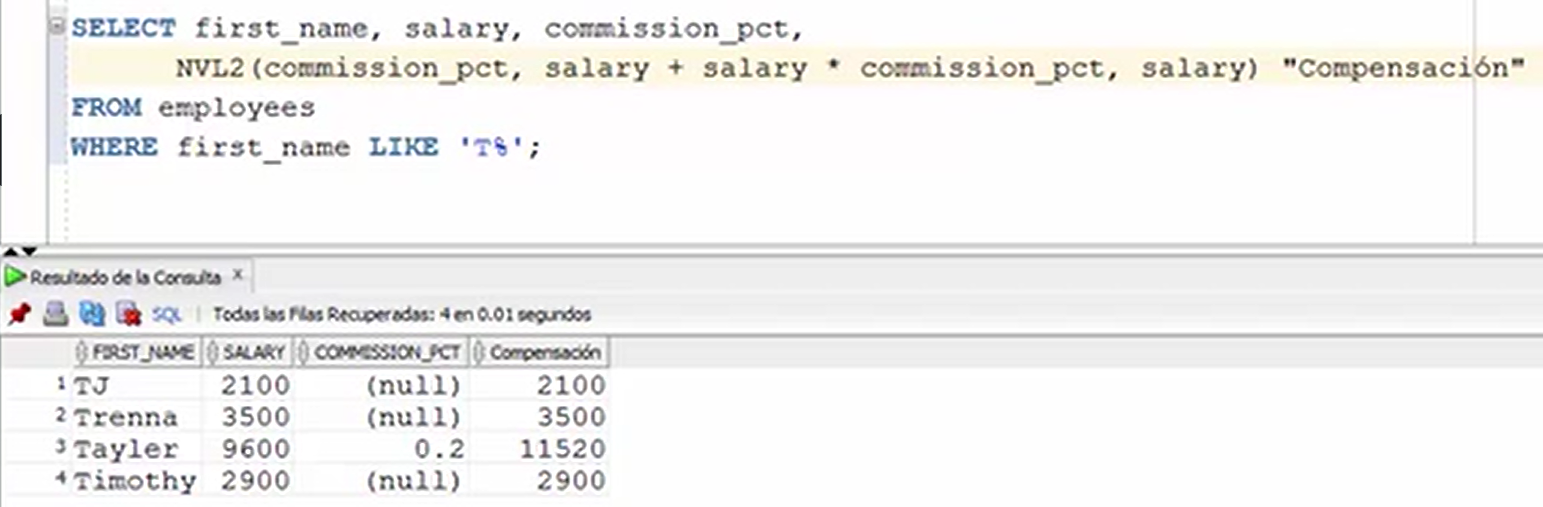


## NVL

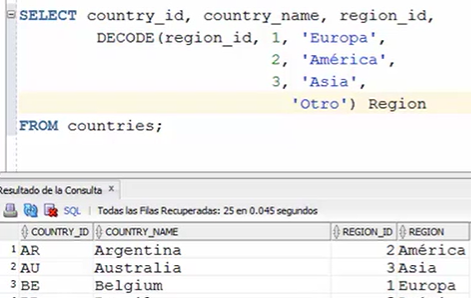




## NVL2

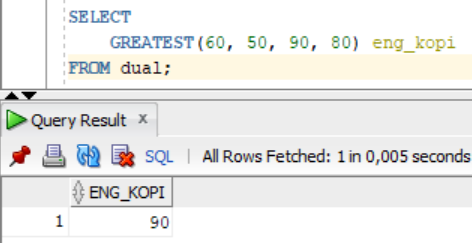


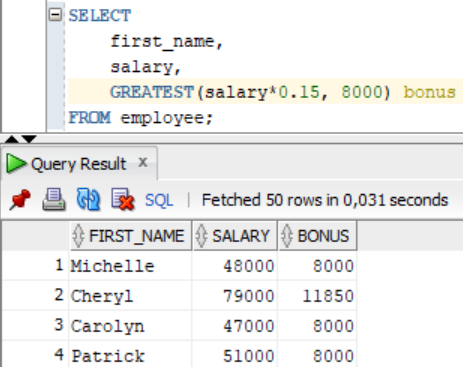
## DECODE()



## GREATEST()

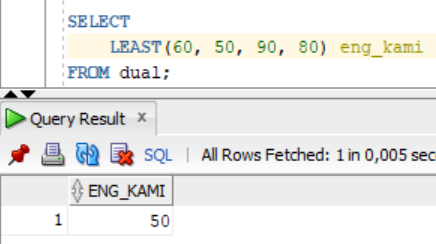
Eng kattasini chiqaradi:

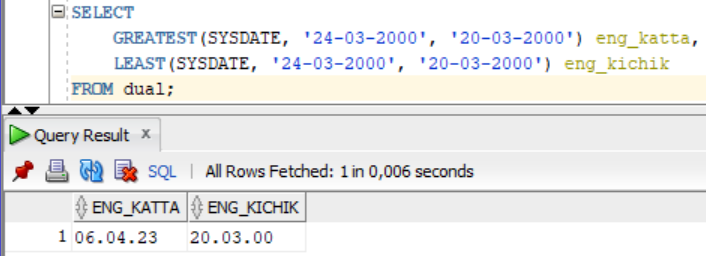




## LEAST()

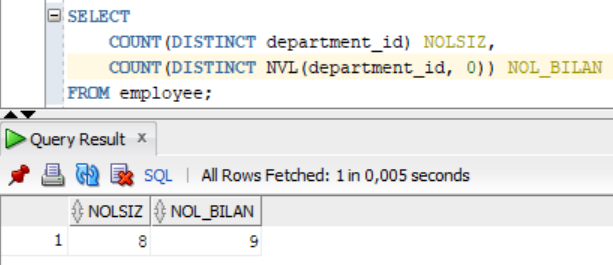
Eng kamini chiqaradi:



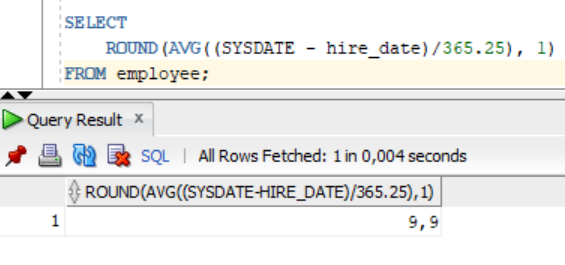


# Group By

## COUNT()

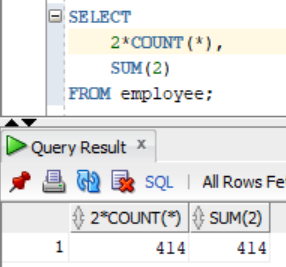


## AVG()

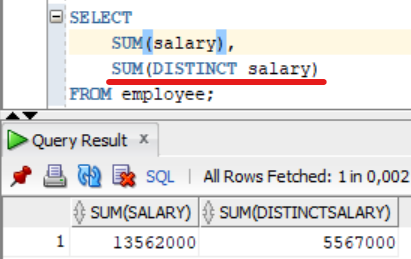


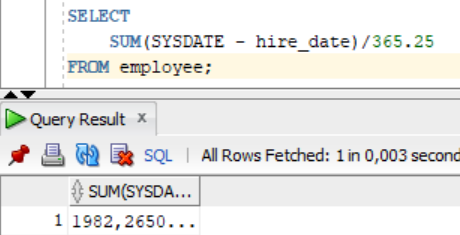
## SUM()

Ikkovi sinonim:

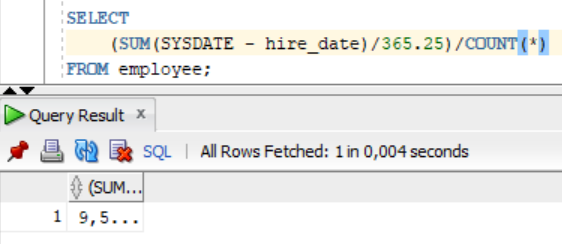


Sum( Distinct salary )



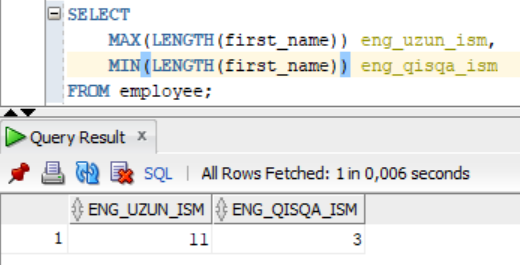


Hodimlarni o`rtacha ishlash yili:



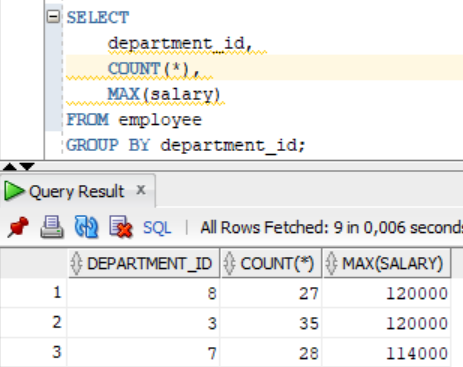
## MIN(), MAX()

Max va Min uzunlikdagi ismlarni topish:

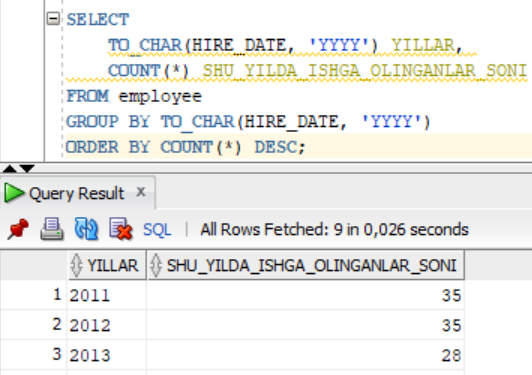


## GROUP BY

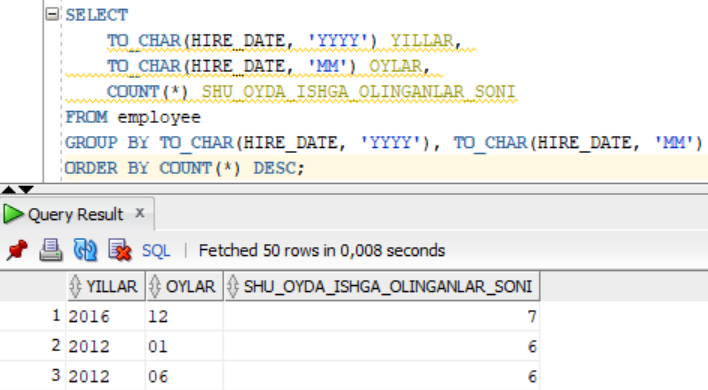
Har bir departament bo`yicha maksimal *salary*ni chiqarish:



Yil bo`yicha ishga olinganlar sonini topish:

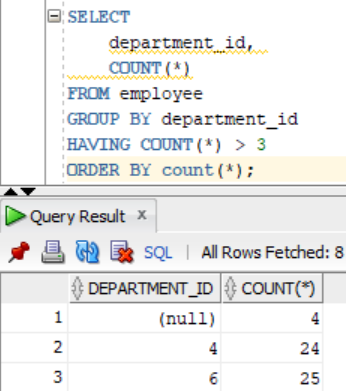


Oy bo`yicha ishga olinganlar sonini topish:

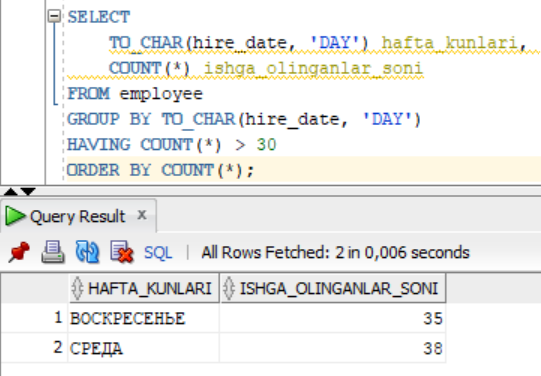


## HAVING bandi

3tadan ko`p hodimi bo`lgan Department\_id ga mansub hodimlar sonini chiqarish:



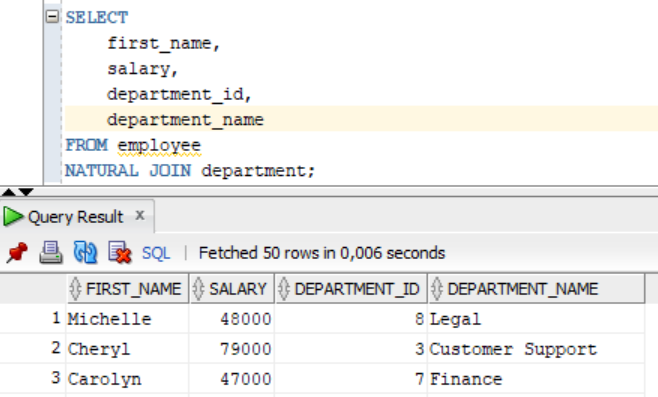
Soni 30 dan oshiq bo`lgan hafta kunlari bo`yicha hodimlar sonini chiqarish:



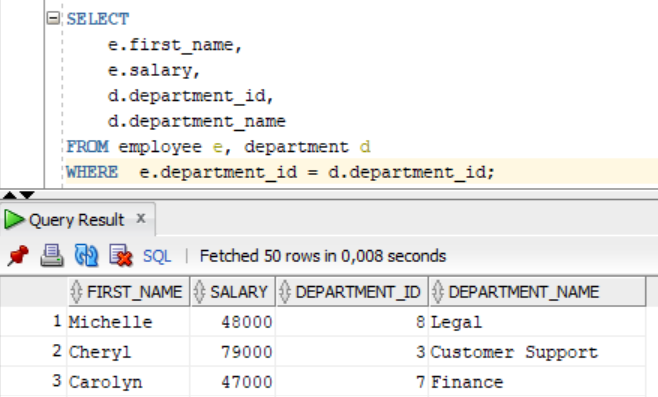
# JOIN

## 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.



Bunga sinonim:

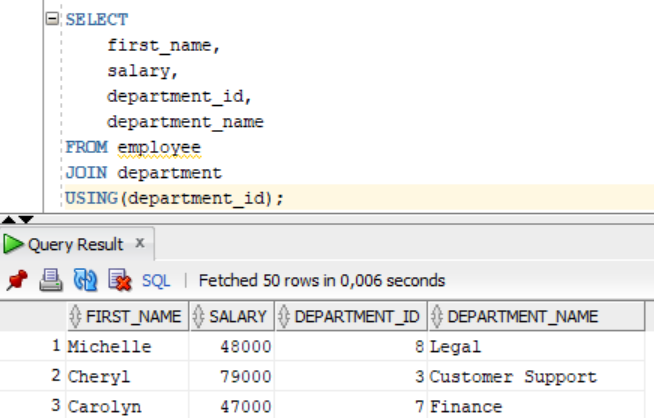


## JOIN

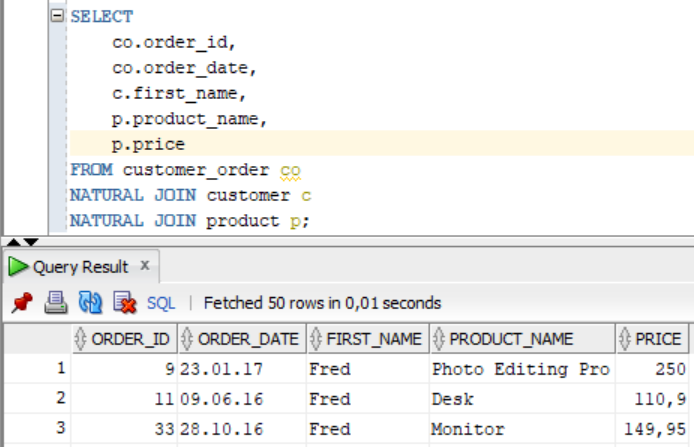
USING ⬄ WHERE e.department\_id = d.department\_id;

USING o`xshash ustunlar nomini kiritish uchun ishlatiladi.

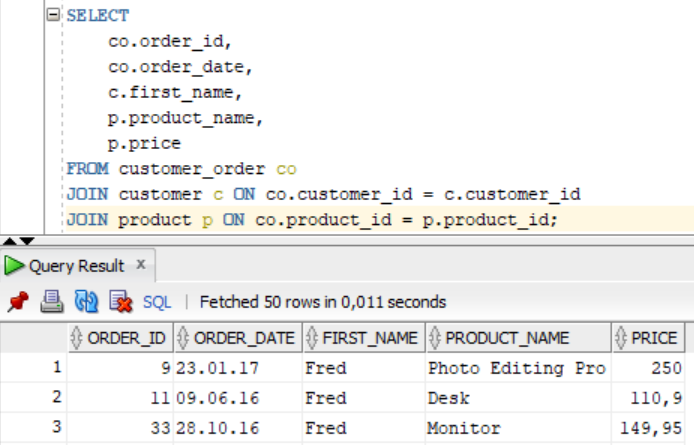
USING ishlatilganda jadvallarga taxallus qo`yilsa xatolik chiqaradi.



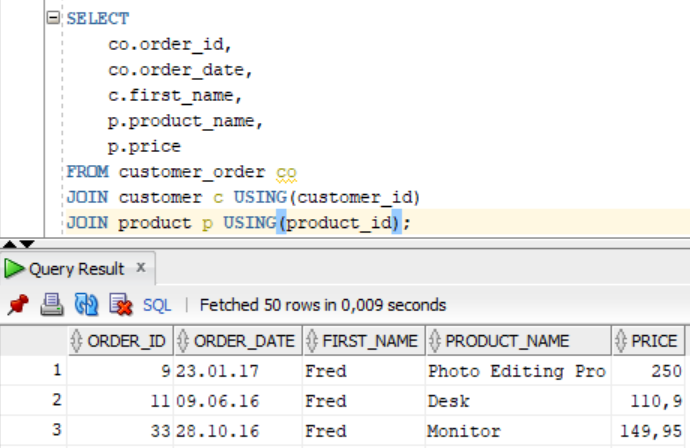
## JOINing Multiple Tables



Bunga sinonim:

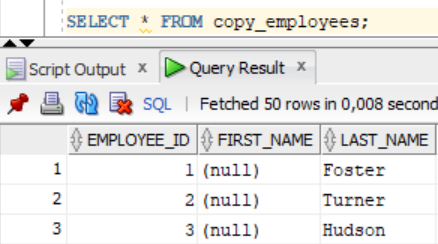
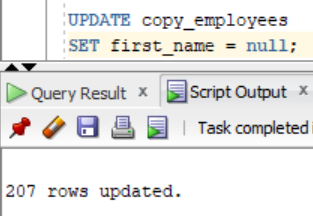


Yana bitta sinonim:

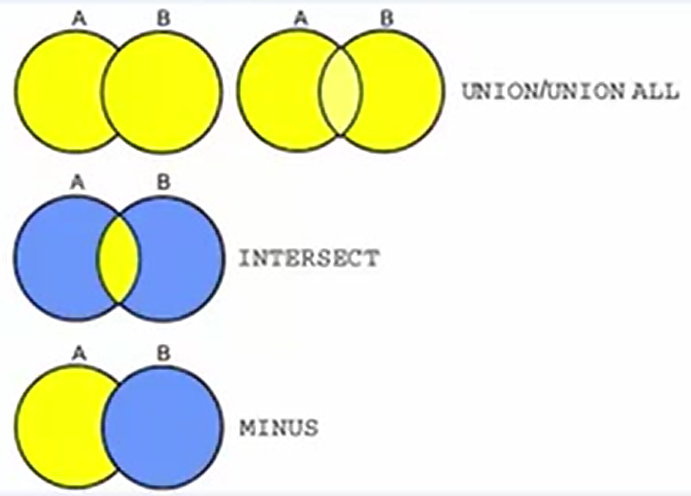


# Insert, Update, Delete

## DML Sentence

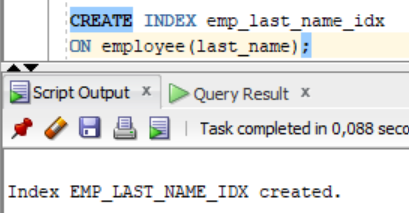


## Operator



## Index

Indeks qatorlarni qidirishni tezlashtiradi.



## 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:

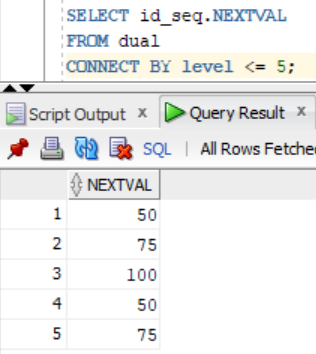
DROP SEQUENCE sequence\_name;

Ushbu SELECT bayonot id\_seq.NEXTVAL qiymatni qayta-qayta ishlatadi:

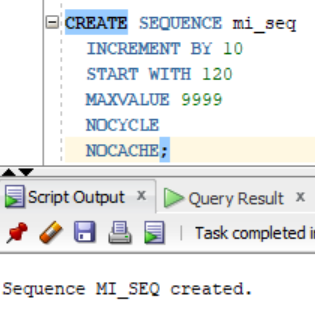
SELECT id\_seq.NEXTVAL

FROM dual

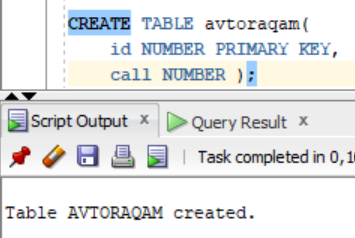
CONNECT BY level <= 5;



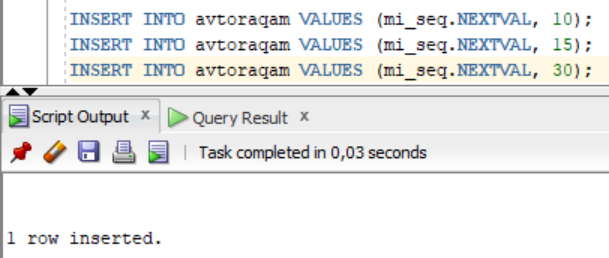
Sequence yaratildi:



Jadval yaratildi:



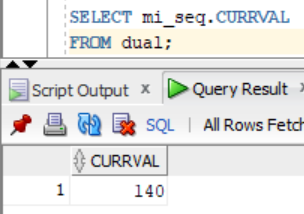
Jadvalga ma’lumotlar kiritildi:



Sequenceni tekshiramiz:



Hozirgi qiymatni aniqlaymiz:

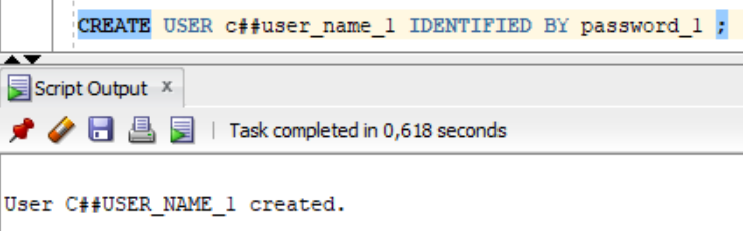


# Sxemalar

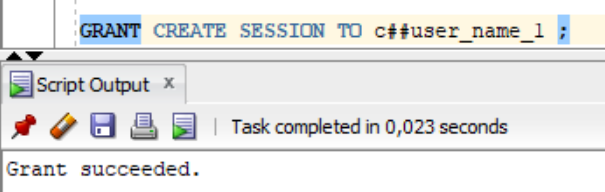
User yaratish:

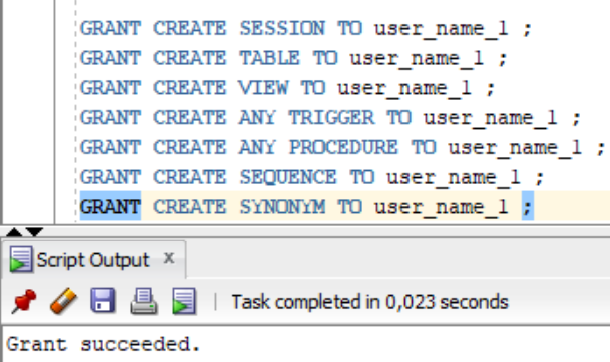


Nomini rad etish holati kuzatilsa:



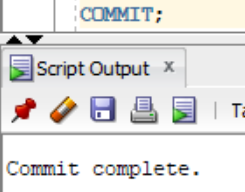
USER larga imtiyoz berish: GRANT



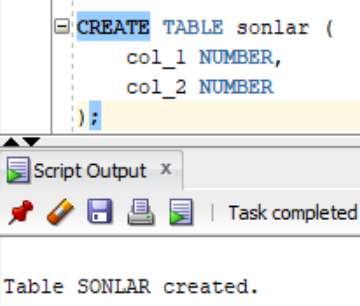
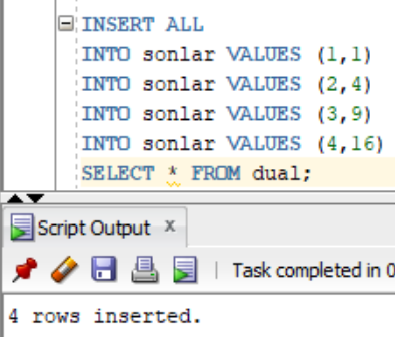
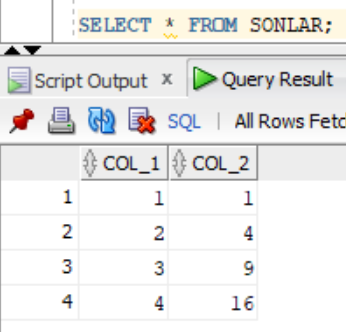


# Transaction

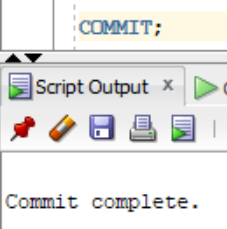
Tranzaksiyada topshiriq boshlanishi/tugashini anglatadi: *COMMIT*



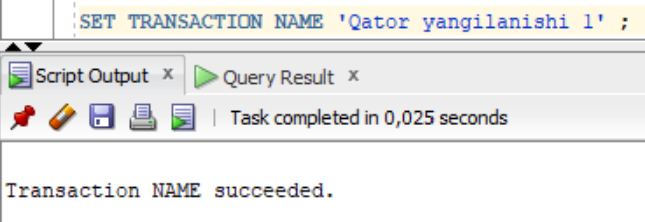
Jadval yaratib, qiymatlar kiritamiz va tekshiramiz:

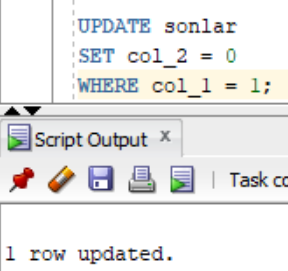
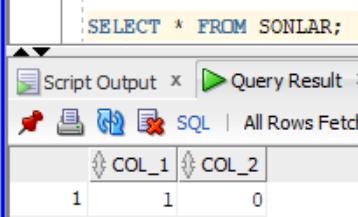
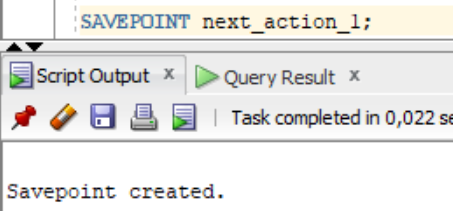
Tranzaksiya topshiriqlarini tugatish:



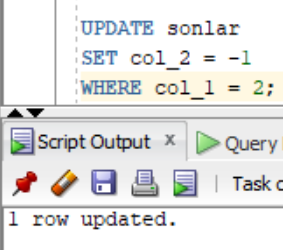
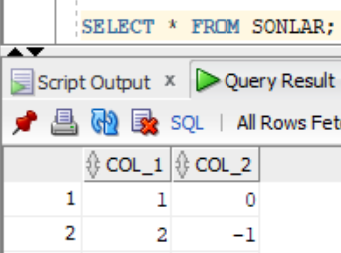
Tranzaksiya o`rnatish: *SET TRANSACTION*



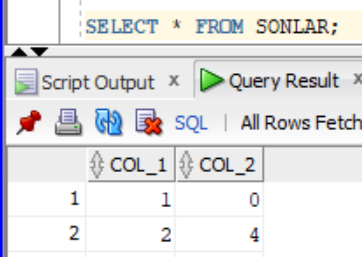
Yangilanish kiritib, tekshiramiz va *SAVEPOINT* ni belgilaymiz:

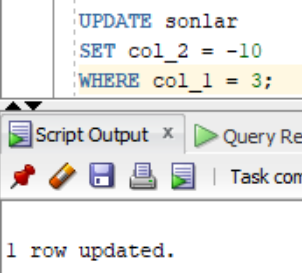
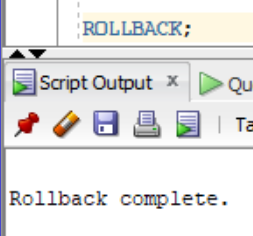
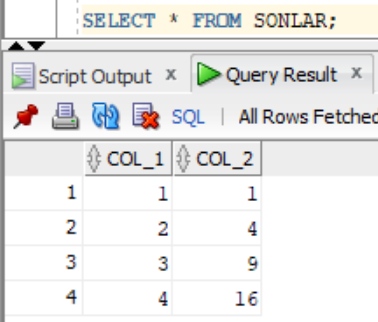
Yangilanish kiritib, tekshiramiz va SAVEPOINT 2-sini belgilaymiz:

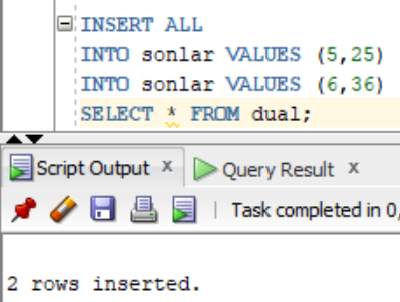
1-SAVEPOINTga qaytamiz va tekshiramiz:



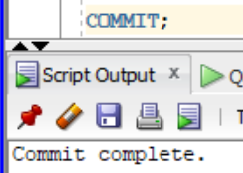
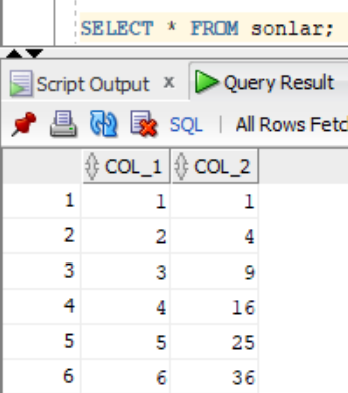
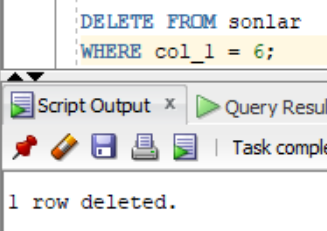
Yangilanish kiritib, boshlang`ich holatga qaytaramiz va tekshiramiz:

Yangilanish kiritamiz:



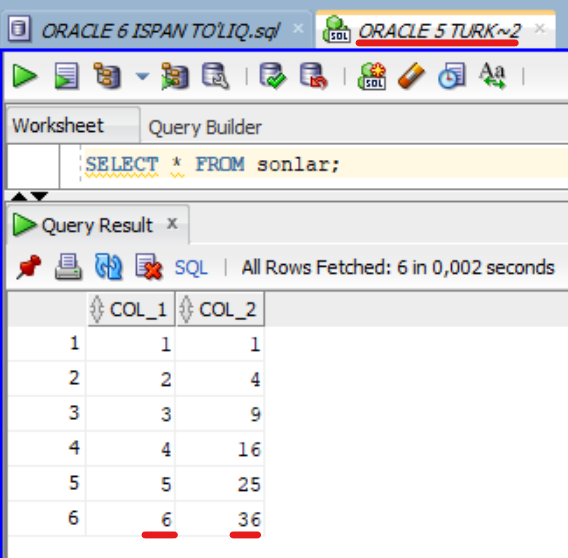
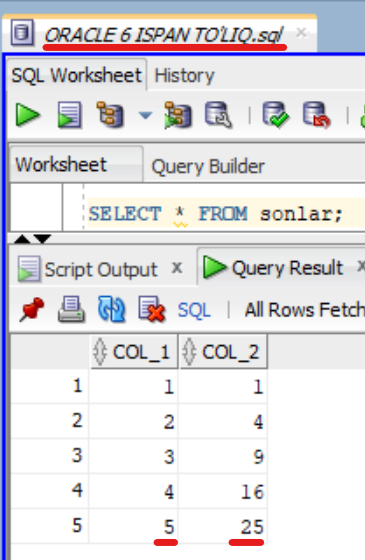
COMMIT qilib, tekshiramiz va DELETE qilamiz:

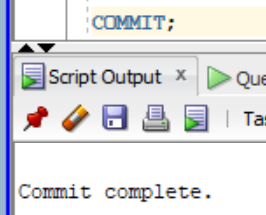
Ispan bazasini va Turk bazasini tekshiramiz:

Natija. Ispanda bajarilgan DELETE Turk bazasida amalga oshirilmagan.

Sabab. Tranzaksiya hali to`liq yakunlanmadi.

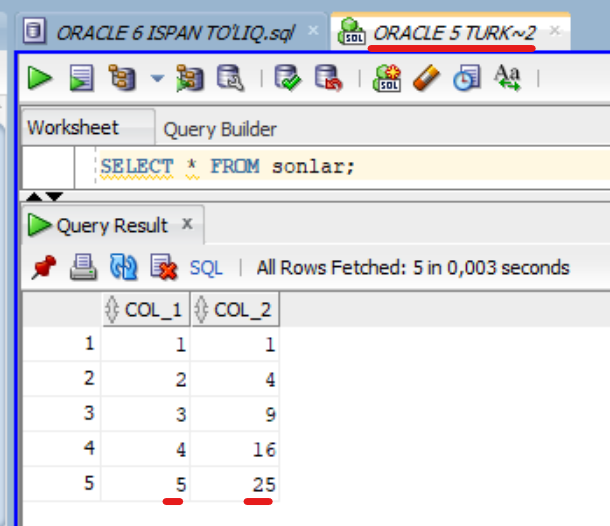


To`liq yakunlash uchun COMMIT qilamiz:



Endi esa Turk bazasida tekshiramiz:

Natija DELETE vazifasi bajarilgan.



# 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;

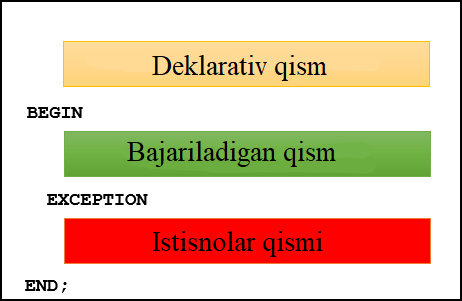
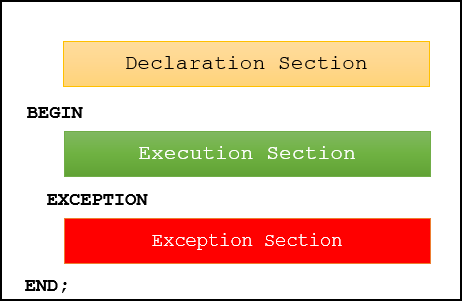
## Indeksni o`chirish

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](https://www.oracletutorial.com/plsql-tutorial/plsql-variables/) , [kursorlar](https://www.oracletutorial.com/plsql-tutorial/plsql-cursor/) uchun xotira ajratadigan va ma’lumotlar turlarini aniqlaydigan deklaratsiya bo`limi mavjud. Bu qismda siz  [variables](https://www.oracletutorial.com/plsql-tutorial/plsql-variables/) ([o`zgaruvchilar](https://www.oracletutorial.com/plsql-tutorial/plsql-variables/) ), [constants](https://www.oracletutorial.com/plsql-tutorial/plsql-constants/) ([doimiylar](https://www.oracletutorial.com/plsql-tutorial/plsql-constants/) ), [cursors](https://www.oracletutorial.com/plsql-tutorial/plsql-cursor/) 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.](https://www.oracletutorial.com/plsql-tutorial/plsql-null/)

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

PL/SQL blokida [EXCEPTION](https://www.oracletutorial.com/plsql-tutorial/plsql-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);