1. MySQL ma’lumotlar bazasini boshqarish tizimi haqida;

MySQL — bu ochiq kodli relyatsion ma’lumotlar bazasini boshqarish tizimi (RDBMS) bo‘lib, u ma’lumotlarni boshqarish va saqlash uchun **SQL (Structured Query Language)** dasturlash tilidan foydalanadi. MySQL keng tarqalgan va ko‘plab veb-ilovalarda, tizimlarda, hamda dasturlarda foydalaniladi. U yuqori tezlik, ishonchlilik va qulaylik kabi jihatlari bilan mashhur. MySQL yordamida ma’lumotlar jadvallar ko‘rinishida saqlanadi va ular ustida qo‘shish, o‘chirish, o‘zgartirish, va qidirish kabi amallar bajariladi. Ushbu tizim ko‘p sonli foydalanuvchilarni qo‘llab-quvvatlaydi va katta hajmdagi ma’lumotlar bilan samarali ishlash imkonini beradi. MySQL bepul bo‘lib, Windows, Linux va MacOS kabi turli operatsion tizimlarda ishlaydi. U ko‘plab dasturlash tillari — masalan, PHP, Python, Java, va C# bilan integratsiyalashadi. MySQL odatda veb-saytlar, internet do‘konlar, mobil ilovalar va korporativ axborot tizimlarida ma’lumotlar bazasi sifatida qo‘llaniladi.

2. MySQL Workbench ni o’rnatish, ishga tushirish va server sozlamalari;

MySQL Workbench — bu MySQL ma’lumotlar bazasi bilan ishlash uchun vizual muhit bo‘lib, u orqali ma’lumotlar bazasini loyihalash, so‘rovlar yozish, ma’lumotlarni boshqarish va server sozlamalarini o‘zgartirish mumkin. Quyida uni o‘rnatish, ishga tushirish va server sozlamalarini bajarish bosqichma-bosqich, aniq ko‘rsatmalar bilan keltirilgan:

---

1. MySQL Workbench’ni o‘rnatish

Windows uchun:

1. https://dev.mysql.com/downloads/workbench/ manziliga o‘ting.
2. “Select Operating System” bo‘limidan **Windows** tanlang.
3. “Download” tugmasini bosing va .msi faylini yuklab oling.
4. Yuklab olingan faylni ishga tushiring va o‘rnatish ustasiga (Installer) amal qiling.
5. O‘rnatish davomida MySQL Server bo‘lmasa, u ham birga o‘rnatiladi (yoki alohida o‘rnating).
6. MySQL rasmiy saytiga o‘ting va macOS versiyasini tanlang.
7. .dmg faylni yuklab oling va ilovani “Applications” papkasiga ko‘chiring.
8.

---

9. **2. MySQL Workbench’ni ishga tushirish**
10. O‘rnatish tugaganidan so‘ng, ish stolidagi yoki Start menyusidagi **MySQL Workbench** belgisini bosing.

11. Dastur ochiladi va asosiy oynada mavjud MySQL ulanishlarini ko‘rasiz.

3. MySQL da DDL komandalari;

MySQL'da DDL komandalar (Data Definition Language – Ma’lumotlar Tuzilmasini Belgilovchi Buyruqlar) ma’lumotlar bazasining tuzilmasini yaratish, o‘zgartirish va o‘chirish uchun ishlatiladi. Quyida **asosiy DDL buyruqlari** va ularning vazifalari aniq tarzda keltirilgan:

1. **`CREATE` – Yaratish**

```
CREATE TABLE talabalar (
    id INT PRIMARY KEY,
    ism VARCHAR(50),
    yosh INT
);
```

2.ALTER – O‘zgartirish
3. DROP – O‘chirish
4. RENAME – Nomini o‘zgartirish

4. MySQL da DML komandalari;

MySQL'da DML komandalar (Data Manipulation Language – Ma’lumotlar Bilan Ishlash Buyruqlari) ma’lumotlar bazasidagi ma’lumotlarni kiritish, o‘zgartirish, o‘chirish va qidirish uchun ishlatiladi. Quyida **DML buyruqlari** va ularning vazifalari aniq tarzda keltirilgan:

Buyruq	Vazifasi
INSERT	Jadvalga yangi ma’lumot kiritish
UPDATE	Jadvaldagi ma’lumotlarni yangilash
DELETE	Jadvaldan ma’lumotlarni o‘chirish
SELECT	Jadvaldan ma’lumotlarni tanlash (qidirish)

```
INSERT INTO talabalar (id, ism, yosh)
VALUES (1, 'Ali', 22);
INSERT INTO talabalar (id, ism, yosh)
VALUES (2, 'Nodira', 23), (3, 'Olim', 21);
UPDATE talabalar
SET yosh = 24
WHERE id = 1;
DELETE FROM talabalar
WHERE id = 1;
```

DELETE FROM talabalar;

5. MySQL da TSL komandalari;

MySQL'da TSL komandalar (Transaction Control Language – Tranzaksiyalarni Boshqarish Buyruqlari) ma’lumotlar bazasida tranzaksiyalarni boshqarish uchun ishlatiladi. Tranzaksiya — bu bir yoki bir nechta SQL buyruqlari to‘plamidir, ularni amalga oshirishda barcha buyruqlar yoki ularning hech biri bajarilishi kerak. TSL komandalarining asosiy maqsadi ma’lumotlar bazasining integritetini ta'minlash va tranzaksiyalarni boshqarishdir.

1. `START TRANSACTION` – Tranzaksiyani boshlash

Funktsiyasi: Yangi tranzaksiya boshlash uchun ishlatiladi. Bu buyruq, tranzaksiya ichidagi barcha SQL buyruqlarini birlashtiradi va ular faqat tranzaksiya muvaffaqiyatli yakunlanganda saqlanadi.

Buyruq	Vazifasi
START TRANSACTION	Tranzaksiyani boshlash
COMMIT	Tranzaksiyani tasdiqlash, o‘zgarishlarni saqlash
ROLLBACK	Tranzaksiyani bekor qilish, o‘zgarishlarni qaytarish
SAVEPOINT	Tranzaksiyada saqlash nuqtasini belgilash
RELEASE SAVEPOINT	Saqlash nuqtasini olib tashlash
SET AUTOCOMMIT	Avtomatik tasdiqlashni yoqish yoki o‘chirish

6. MySQL da ma’lumotlar tiplari;

MySQL'da ma'lumotlar tiplari — bu ma'lumotlar bazasida saqlanadigan qiymatlarning turini belgilaydi. Har bir ustun uchun tegishli ma'lumot tipi tanlanishi kerak, chunki bu ma'lumotni saqlash va ishlov berishda muhim ahamiyatga ega. MySQL da ma'lumotlar tiplari asosan uch asosiy guruhga bo‘linadi:

- **Matnli ma'lumotlar (String types)**
- **Sana va vaqt (Date and Time types)**
- **Binar ma'lumotlar (Binary types)**
- **CHAR**: Belirli uzunlikdagi matn. CHAR(n) deb yoziladi, bu n belgilangan uzunlikka ega bo‘lgan matnni saqlaydi. Agar kiritilgan matn uzunligi n dan kam bo‘lsa, qolgan joylar bo‘shliq bilan to‘ldiriladi.

- **Misol:** kod CHAR(5)

- **VARCHAR**: O‘zgaruvchan uzunlikdagi matn. Bu tipda matnning uzunligi saqlanadi, ya'ni faqat haqiqiy belgilangan uzunlik saqlanadi.
- **DATE**: Sana (yil, oy, kun) ma'lumotlarini saqlaydi. Format: YYYY-MM-DD.

- **Misol:** tugilgan_sana DATE ˅ **DATETIME**: Sana va vaqtni saqlaydi. Format: YYYY-MM-DD HH:MM:SS.**Misol:** yaratish_vaqti DATETIME

7. MySQL da PrimaryKey cheklovi. Misollar keltiring;

MySQL'da PRIMARY KEY cheklovi — bu ma’lumotlar bazasida har bir jadval uchun noyob identifikatorni ta’minlash maqsadida qo‘llaniladi. PRIMARY KEY cheklovi, jadvaldagi har bir qatorni (yoki yozuvni) yagona va farqlashga yordam beradi. Har bir jadvalda faqat bitta PRIMARY KEY bo‘lishi mumkin. PRIMARY KEY quyidagi xususiyatlarga ega:

- **Noyoblik**: PRIMARY KEY bo‘lishi kerak bo‘lgan ustundagi har bir qiymat yagona (unique) bo‘lishi kerak. Ya'ni, bir xil qiymat takrorlanmasligi kerak.
- **Not NULL**: PRIMARY KEY bo‘lgan ustundagi har bir qiymat **NULL bo‘lmasligi** kerak. Bu, har bir yozuv uchun noyob identifikator bo‘lishi uchun zarurdir.
- **Index**: PRIMARY KEY avtomatik ravishda indeks sifatida yaratiladi, bu esa ma'lumotlarga tezkor kirishni ta’minlaydi.

```
CREATE TABLE talabalar (
    id INT NOT NULL,
    ism VARCHAR(50),
    yosh INT,
    PRIMARY KEY (id)
);
```

8. MySQL da ForeignKey cheklovi. Misollar keltiring;

MySQL’da FOREIGN KEY cheklovi — bu ma'lumotlar bazasida bir jadvaldagi ma'lumotlarni boshqa jadvaldagi ma'lumotlar bilan bog‘lash uchun ishlatiladi. FOREIGN KEY ma'lumotlar bazasida **referensial yaxlitlik**ni ta'minlashga yordam beradi, ya'ni bir jadvaldagi qiymatlarning boshqa jadvalda mavjud bo‘lgan qiymatlar bilan o‘zaro bog‘lanishini ta'minlaydi. Bu cheklov **ikki jadval o‘rtasida bog‘lanish** o‘rnatadi, bunda bir jadvalda bir ustun (chetlashgan kalit) ikkinchi jadvaldagi bir ustunga (asosiy kalit) bog‘lanadi.

FOREIGN KEY cheklovi asosiy xususiyatlari

1. **Referensial yaxlitlikni ta'minlash**: FOREIGN KEY jadvaldagi ma'lumotlarni boshqasiga bog‘lash va ular o‘rtasidagi yaxlitlikni ta'minlashga yordam beradi.
2. **Chet el kaliti**: FOREIGN KEY ustuni boshqa jadvaldagi PRIMARY KEY yoki UNIQUE ustuniga bog‘lanadi.

   ON DELETE va **ON UPDATE** cheklovlari: FOREIGN KEY bilan birga **on delete** va **on update** amallari qo‘llanishi mumkin, bu esa chet el kalitiga ta’sir qiladigan holatlarni nazorat qilishga imkon beradi. CREATE TABLE buyurtmalar (buyurtma_id INT NOT NULL,talaba_id INT,sana DATE, PRIMARY KEY (buyurtma_id),

FOREIGN KEY (talaba_id) REFERENCES talabalar(id)
ON DELETE CASCADE ON UPDATE CASCADE)

9. MySQL CHEK cheklovi. Misollar keltiring;

MySQL'da CHECK cheklovi — bu ma'lumotlar bazasidagi ustunlarga kiritiladigan ma'lumotlarning ma'lum bir shartlarga javob berishini ta'minlash uchun ishlatiladi. CHECK cheklovi yordamida ustunlarda saqlanadigan qiymatlarning to‘g‘riligini nazorat qilish mumkin. Masalan, ma'lum bir qiymat diapazonida bo‘lishi yoki ba'zi shartlarga mos kelishi kerak bo‘lishi mumkin.

CHECK cheklovining asosiy xususiyatlari:

- CHECK cheklovi ustunlarga kiritilgan qiymatlarning mosligini tekshiradi.
- CHECK cheklovi qiymatlarning miqdori yoki formatini nazorat qilishi mumkin (masalan, sanani tekshirish, ma'lum raqamlar diapazonini cheklash va h.k.).

```
CREATE TABLE talabalar (
    id INT NOT NULL,
    ism VARCHAR(50),
    yosh INT,
    PRIMARY KEY (id),
    CHECK (yosh >= 18 AND yosh <= 100)
);
```

10. MySQL Autoincrement komandasi. Misollar keltiring;

MySQL’da AUTO_INCREMENT komandasi — bu ma'lumotlar bazasida ustunning qiymatini avtomatik tarzda oshirish uchun ishlatiladi. AUTO_INCREMENT yordamida, bir jadvalga yangi yozuv qo‘shilganida, bu ustun uchun qiymat avtomatik tarzda hisoblanadi. Bu asosan, ma'lumotlar bazasida **unikal identifikatorlar** (masalan, id ustuni) yaratishda qo‘llaniladi.

AUTO_INCREMENT haqida umumiy ma'lumot

- **Avtomatik oshish**: Har safar yangi yozuv qo‘shilganda, AUTO_INCREMENT ustunidagi qiymat avvalgi qiymatdan 1 ga oshadi. Bu orqali har bir yozuv uchun unikallik ta'minlanadi.
- **Birinchi qiymat**: AUTO_INCREMENT qiymati 1 dan boshlanadi (lekin boshlang‘ich qiymatni o‘zgartirish mumkin).
- **NULL qiymatlari**: AUTO_INCREMENT ustuniga qiymat kiritish shart emas. Agar NULL yoki qiymat kiritilmasa, MySQL avtomatik ravishda yangi qiymatni tayinlaydi.

```
CREATE TABLE talabalar (
    id INT AUTO_INCREMENT,
    ism VARCHAR(50),
    yosh INT,
    PRIMARY KEY (id)
```

-);

11. Bazadan yozuvlar tanlash (Select). AND, OR va NOT operatorlari;

MySQL’da yozuvlarni tanlash (SELECT) — bu SQL so'rovi orqali ma'lumotlar bazasidagi ma'lumotlarni tanlash va ko'rsatish jarayonidir. SELECT operatori yordamida bazadagi jadvaldan ma'lumotlarni filtrlash, saralash va tanlash mumkin. Bu jarayonda **AND**, **OR**, va **NOT** operatorlari yordamida ma'lumotlarni yanada aniqroq va shartlarga mos ravishda tanlash mumkin.

SELECT ism, yosh FROM talabalar;

SELECT ism, yosh
FROM talabalar
WHERE yosh > 20 AND ism = 'Ali';

Bu so‘rovda:

- yosh > 20 va ism = 'Ali' shartlari AND operatori bilan birlashtirilgan.
- Shartlar ikkalasi ham **true** bo‘lishi kerak, ya'ni yosh 20 dan katta bo‘lishi va ismi "Ali" bo‘lishi kerak.

SELECT ism, yosh
FROM talabalar

- WHERE yosh > 20 OR ism = 'Ali';
  - yosh > 20 yoki ism = 'Ali' shartlaridan bittasi **true** bo‘lsa, natija qaytariladi.
  - Demak, yosh 20 dan katta bo‘lgan talabalar yoki ismi "Ali" bo‘lgan talabalar ko‘rsatiladi.

12. Bazadan yozuvlar tanlash (Select). NULL operatori.

MySQL’da SELECT operatori yordamida yozuvlar tanlash — bu ma'lumotlar bazasidan ma'lumotlarni so‘rab olish jarayonidir. MySQL’da **NULL** qiymatlari bilan ishlash muhim ahamiyatga ega, chunki NULL qiymatlari ma'lum bir ustunda qiymatning mavjud emasligini yoki noma'lumligini bildiradi. MySQL’da NULL qiymatlar bilan ishlashda o‘ziga xos operatorlar va shartlar mavjud.

SELECT ism, yosh FROM talabalar;

NULL Qiymatlari Bilan Ishlash

NULL — bu ma'lumotlar bazasida qiymatning mavjud emasligini yoki noma'lumligini bildiradi. MySQL’da NULL qiymatlari bilan ishlashda IS NULL va IS NOT NULL operatorlari ishlatiladi. Shuningdek, NULL qiymatlar uchun = yoki != operatorlari ishlatilmaydi, chunki NULL bilan taqqoslash **har doim noto‘g‘ri** (false) natija beradi.

SELECT ism, telefon

FROM talabalar WHERE telefon IS NULL;

13. MySQL da satrlar bilan ishlash funksiyalari: ASCII(), CHAR_LENGTH(), CHARACTER_LENGTH();

MySQL'da **satrlar bilan ishlash funksiyalari** satrlarni analiz qilish, o'zgartirish va manipulyatsiya qilish uchun ishlatiladi. Bu funksiyalar ma'lumotlar bazasidagi satrlarni turli usullar bilan ko'rib chiqish va boshqarish imkonini beradi. Quyida keltirilgan funksiyalar: **ASCII()**, **CHAR_LENGTH()**, va **CHARACTER_LENGTH()** — ular satrlar bilan ishlashda eng ko'p ishlatiladigan funksiyalardan biridir.

SELECT ASCII('A');
SELECT ASCII('a');

CHAR_LENGTH() funksiyasi satrning belgilari sonini hisoblaydi. Bu funksiya satrning uzunligini belgilarning umumiy soni sifatida qaytaradi va bu har bir belgining o'lchamini hisobga oladi. Unicode belgilarini ham to'liq hisoblaydi. SELECT CHAR_LENGTH('Hello World'); -- 11

CHARACTER_LENGTH()

CHAR_LENGTH() bilan bir xil ishlaydi, satrning belgilari sonini hisoblaydi.

14. MySQL da satrlar bilan ishlash funksiyalari: CONCAT(), FIELD();

MySQL'da satrlar (matnlar) bilan ishlash funksiyalari turli matnlarni birlashtirish, taqqoslash yoki analiz qilishda yordam beradi. Ushbu savolda so'ralgan CONCAT() va FIELD() funksiyalari matnli qiymatlar bilan ishlashda juda foydali bo'ladi.

CONCAT() funksiyasi bir nechta satrlarni **birlashtirish (yopishtirish)** uchun ishlatiladi. U argument sifatida berilgan barcha matnlarni ketma-ket ulab, yagona satr qilib qaytaradi.

SELECT CONCAT(ism, ' ', familiya) AS toliq_ism
FROM talabalar;

FIELD() funksiyasi berilgan qiymatning argumentlar ichida **qaysi o'rinda** turganini qaytaradi (ya'ni indeksini). Agar qiymat topilmasa, **0** qaytaradi.

SELECT ism, FIELD(guruh, 'A', 'B', 'C') AS guruh_darajasi
FROM talabalar;

SELECT FIELD('nok', 'banan', 'olma', 'shaftoli') AS natija;

15. MySQL da satrlar bilan ishlash funksiyalari: FIND_IN_SET(), FORMAT();

FIND_IN_SET() funksiyasi biror **qiymat**ning vergul bilan ajratilgan **satr ichida qaysi o‘rinda** (nechanchi pozitsiyada) turganini qaytaradi. Qiymat topilsa, indeks (raqam) sifatida, topilmasa 0 qaytaradi.

```
SELECT nomi
FROM mahsulotlar
WHERE FIND_IN_SET('telefon', turlari) > 0;
SELECT FIND_IN_SET('olma', 'banan,olma,shaftoli');
```

FORMAT() funksiyasi sonlarni **belgilangan kasr soni bilan** formatlaydi va **minglik ajratkichlar** bilan satr sifatida qaytaradi. Ko‘proq **hisob-kitob natijalarini chiroyli ko‘rsatish** uchun ishlatiladi.

```
SELECT nomi, FORMAT(narx, 2) AS chiroyli_narx
FROM mahsulotlar;
SELECT FORMAT(1234567.891, 2);
```

16. MySQL da satrlar bilan ishlash funksiyalari: INSERT(), INSTR();

INSERT() funksiyasi satr ichiga boshqa satrni **ma'lum bir pozitsiyadan boshlab qo‘yish (joylashtirish)** uchun ishlatiladi.

INSERT(original_string, start_position, length, insert_string)

- original_string — asosiy matn (qayerga qo‘yilishini istaysiz).
- start_position — qaysi belgidan boshlab qo‘yiladi (1 dan boshlanadi).
- length — necha belgini almashtirish kerak.
- insert_string — qo‘yiladigan matn.

```
SELECT INSERT('Salom dunyo', 7, 5, 'O‘zbekiston');
```

INSTR() funksiyasi biror **kichik satr (substring)** asosiy satr ichida **birinchi bor qayerda uchrashini** aniqlaydi (indeksini qaytaradi).

INSTR(original_string, search_string)

original_string — ichida qidiriladigan satr.

search_string — qidirilayotgan kichik matn.

```
SELECT INSTR('Salom dunyo', 'dun');
```

17. MySQL da satrlar bilan ishlash funksiyalari: LCASE(), LOWER(), LEFT() va RIGHT();

LCASE() funksiyasi satrdagi **barcha harflarni kichik harfga** aylantiradi.

SELECT LCASE('SALOM Dunyo');

LOWER() funksiyasi ham LCASE() bilan **bir xil ishlaydi**. Har ikkisi ham satrdagi barcha harflarni kichik holatga o‘zgartiradi.

SELECT LOWER('MySQL DASTURI');

LEFT() funksiyasi satrning **chap tomonidan** belgilangan miqdorda belgilarni ajratib oladi.

LEFT(satr, n)

satr: Matnli qiymat.

n: Nechta belgini olish kerakligi.

Misollar:

SELECT LEFT('O‘zbekiston', 5);

RIGHT() funksiyasi satrning **o‘ng tomonidan** belgilangan miqdorda belgilarni ajratib oladi.

SELECT RIGHT('O‘zbekiston', 6);

18. MySQL da satrlar bilan ishlash funksiyalari: LENGTH(), LOCATE();

LENGTH() funksiyasi satrdagi **baytlar sonini** qaytaradi (harflar emas!).

- ASCII belgilar uchun 1 bayt = 1 belgi.
- Unicode (masalan, kirill yoki arabcha) belgilar ko‘proq bayt egallashi mumkin.

SELECT LENGTH('Hello');

LOCATE() funksiyasi kichik satrning asosiy satr ichida qayerda (nechanchi pozitsiyada) joylashganini aniqlaydi. Agar topilmasa, 0 qaytaradi.

LOCATE(substring, string [, start_position])

substring – izlanayotgan kichik matn.

string – asosiy matn.

start_position – ixtiyoriy. Qaysi pozitsiyadan izlashni boshlash.

SELECT LOCATE('o', 'Salom dunyo', 6);

9 — 6-pozitsiyadan boshlab qidirganda 'o' 9-pozitsiyada uchraydi.

19. MySQL da satrlar bilan ishlash funksiyalari: MID(), POSITION();

MID() funksiyasi satrning ichidan belgilangan joydan boshlab, belgilangan uzunlikdagi qismini ajratib oladi. Bu funksiya SUBSTRING() bilan bir xil ishlaydi.

MID(satr, boshlanish_pozitsiyasi, uzunlik)

satr – ishlanadigan matn.

boshlanish_pozitsiyasi – qaysi belgidan boshlab olish kerak (1 dan boshlanadi).

uzunlik – nechta belgi ajratib olinadi.

SELECT MID('O‘zbekiston Respublikasi', 11, 5);

POSITION() funksiyasi **kichik satrning asosiy satr ichidagi birinchi uchrash pozitsiyasini** topadi. Bu LOCATE() funksiyasi bilan juda o‘xshash.

POSITION(substring IN string)

substring – izlanayotgan qism.

string – qayerda izlanadi.

20. MySQL da satrlar bilan ishlash funksiyalari: REPEAT(), REPLACE();

REPEAT() funksiyasi, berilgan **matnni** belgilangan **miktorda takrorlaydi**.

REPEAT(satr, takrorlash_soni)

satr — takrorlanadigan matn.

takrorlash_soni — nechta takrorlash kerakligi.

SELECT nomi, REPEAT(nomi, 2) AS takrorlangan_nomi

FROM mahsulotlar;

Bu so‘rov har bir mahsulot nomini 2 marta takrorlaydi.

REPLACE() funksiyasi, berilgan satrda biror **kichik satrni** (substring) **yangi matn bilan almashtiradi**. REPLACE(satr, eski_qism, yangi_qism)

satr — matn, unda almashtirishlar amalga oshiriladi.

eski_qism — almashtiriladigan qism. yangi_qism — almashtiriladigan yangi matn.

21. MySQL da satrlar bilan ishlash funksiyalari: REVERSE(), RPAD();

REVERSE() funksiyasi berilgan satrni **teskari tartibda** qaytaradi, ya'ni so'ngidan boshlab boshigacha.

SELECT REVERSE('Salom');

SELECT ism, REVERSE(ism) AS teskari_ism FROM talabalar;

Bu har bir ismni teskari holatda ko'rsatadi, masalan: Ali → ilA

RPAD() funksiyasi satrni belgilangan uzunlikkacha **o'ng tomondan to'ldiradi** (padding). Bu funksiyadan **matnni formatlash** uchun foydalaniladi.

RPAD(satr, umumiy_uzunlik, to'ldiruvchi_satr)

satr — asosiy matn.

umumiy_uzunlik — hosil qilinadigan natijaning umumiy uzunligi.

to'ldiruvchi_satr — o'ng tomonga qo'shiladigan matn.

SELECT RPAD(ism, 10, '-') AS formatlangan_ism FROM talabalar;

Natija: Ali-------, Aziza-----

Ismlar 10 belgiga yetguncha - bilan to'ldiriladi.

22. MySQL da satrlar bilan ishlash funksiyalari: SPACE(), STRCMP();

`SPACE()` funksiyasi berilgan son asosida **shuncha bo'sh joy (space)** yaratadi.

SELECT CONCAT('Salom', SPACE(5), 'Dunyo');

'Salom' bilan 'Dunyo' orasida 5 ta bo'sh joy bor.

SELECT CONCAT('Ism:', SPACE(3), 'Javlon');

STRCMP() (String Compare) funksiyasi **ikkita satrni taqqoslaydi** va quyidagicha **raqamli natija** qaytaradi:

- 0 → agar satrlar **bir xil** bo'lsa.
- 1 → agar **birinchi satr katta** bo'lsa (leksik tartibda).
- -1 → agar **ikkinchi satr katta** bo'lsa.

SELECT STRCMP('apple', 'apple');

Natija:0

23. MySQL da satrlar bilan ishlash funksiyalari: SUBSTRING(), SUBSTRING_INDEX();

SUBSTRING() funksiyasi satrdan **belgilangan pozitsiyadan boshlab**, **belgilangan uzunlikdagi qismni ajratib oladi**. Bu funksiya MID() bilan bir xil ishlaydi.**Sintaksis:**

SUBSTRING(satr, boshlanish_pozitsiyasi, uzunlik)

- boshlanish_pozitsiyasi — qaysi belgidan boshlash kerak (1 dan boshlanadi).
- uzunlik — nechta belgi ajratib olish kerak

SELECT SUBSTRING('O‘zbekiston Respublikasi', 13, 5);

13-pozitsiyadan 5 ta belgi ajratib olingan.

SUBSTRING_INDEX() funksiyasi **berilgan ajratgichga (delimiter)** qarab satrni bo‘ladi va **belgilangan qismini qaytaradi**.

SUBSTRING_INDEX(satr, ajratgich, son)

satr — tahlil qilinadigan matn.

ajratgich — bo‘lish belgisi (masalan: ,, . yoki @).

Son:Musbat bo‘lsa: chapdan boshlab necha qism olinadi.

Manfiy bo‘lsa: o‘ngdan boshlab necha qism olinadi.

SELECT SUBSTRING_INDEX('ali@gmail.com', '@', -1);

@ bo‘yicha bo‘linadi, o‘ngdan 1-qism olinadi.

24. MySQL da satrlar bilan ishlash funksiyalari: TRIM(), UPPER(), UCASE();

TRIM() funksiyasi satrning **boshi va oxiridagi bo‘sh joylarni (probel)** yoki **belgilangan belgilarni** olib tashlaydi.

TRIM(satr)
TRIM([LEADING | TRAILING | BOTH] belgilar FROM satr)
SELECT TRIM(BOTH '*' FROM '***Hello***');
Natija: 'Hello'
UPPER() funksiyasi satrdagi **barcha harflarni katta harflarga** aylantiradi.
SELECT UPPER('salom dunyo');
UCASE() — bu UPPER() funksiyasining sinonimi. U ham **matnni katta harflarga** aylantiradi.
SELECT UCASE('mysql dasturi');

25. MySQL da raqamlar bilan ishlash funksiyalari: AVG(), CEIL() va FLOUR();

AVG() funksiyasi bir ustundagi **raqamlar yig‘indisini elementlar soniga bo‘lib, o‘rtacha qiymatni** qaytaradi.

SELECT AVG(ustun_nomi) FROM jadval_nomi;

SELECT AVG(maosh) AS ortacha_maosh FROM ishchilar;

CEIL() (yoki CEILING()) funksiyasi sonni **eng yaqin butun songa yuqoriga qarab** yaxlitlaydi

SELECT CEIL(4.3);

Natija :5

FLOOR() funksiyasi sonni **eng yaqin butun songa pastga qarab** yaxlitlaydi.

SELECT FLOOR(raqam);

SELECT FLOOR(4.7);

Natija :4

26. MySQL da raqamlar bilan ishlash funksiyalari: COUNT(), ROUND() va SUM;

COUNT() funksiyasi jadvaldagi **nechta yozuv mavjudligini** yoki **berilgan ustunda nechta qiymat mavjudligini** hisoblaydi. SELECT COUNT(*) FROM jadval_nomi;

SELECT COUNT(ustun_nomi) FROM jadval_nomi;

SELECT COUNT(*) FROM talabalar;

ROUND() funksiyasi sonni **belgilangan kasr soni bo‘yicha yaxlitlaydi**.

SELECT ROUND(3.14159, 1);

Natija:3.1

SUM() funksiyasi ustundagi barcha sonlar yig‘indisini hisoblaydi.

Sintaksis:

SELECT SUM(ustun_nomi) FROM jadval_nomi;

SELECT SUM(maosh), AVG(maosh) FROM ishchilar;

Bu umumiy yig‘indi va o‘rtacha maoshni bir vaqtning o‘zida qaytaradi.

27. MySQL da raqamlar bilan ishlash funksiyalari: RAND(), TRUNCATE(),

RAND() funksiyasi **0 va 1 orasida** joylashgan **tasodifiy son** (floating-point) hosil qiladi.

RAND()

RAND(seed)

seed — ixtiyoriy butun son. Agar kiritilsa, tasodifiy sonlar har doim bir xil tartibda hosil bo‘ladi (test qilish uchun foydali).

SELECT FLOOR(1 + (RAND() * 100));

Natija: 1–100 oralig‘idagi butun son (masalan, 47)

TRUNCATE() funksiyasi raqamni **belgilangan kasr raqamgacha qirqadi**, ya’ni **yaxlitlamaydi**, faqat kesadi

SELECT TRUNCATE(3.14159, 2);

Natija: 3.14

Bu ROUND()dan farqli. ROUND() 3.15 qiladi, TRUNCATE() esa kesib tashlaydi, ya’ni 3.14.

28. MySQL da sana va vaqt bilan ishlash funksiyalari: ADDDATE(), ADDTIME(;)

ADDDATE() funksiyasi berilgan sanaga **kunlar** (yoki **INTERVAL** orqali oy, yil va boshqalar) qo‘shadi.

Sintaksis:

ADDDATE(sana, kun_soni)

ADDDATE(sana, INTERVAL miqdor birlik)

Masalan.5 kun qo’shish

SELECT ADDDATE('2025-05-14', 5);

SELECT ADDDATE('2025-05-14', INTERVAL 1 YEAR);

SELECT ADDDATE('2025-05-14', INTERVAL 10 DAY);

Natijalar: '2026-05-14' va '2025-05-24'

ADDTIME() funksiyasi **vaqtga yoki sanali vaqtga** (DATETIME) **soat, daqiqa yoki soniya** qo‘shadi.

SELECT ADDTIME('10:30:00', '01:45:00');

Natija: '12:15:00'

29. MySQL da sana va vaqt bilan ishlash funksiyalari: CURDATE(), CURRENT_TIMESTAMP();

CURDATE() funksiyasi hozirgi **sana**ni (yil-oy-kun formatida) qaytaradi.
U **faqat sana** qismini beradi, vaqt qismi yo‘q.
Sintaksis:
SELECT CURDATE();

CURRENT_TIMESTAMP() funksiyasi hozirgi **sana va vaqt**ni (yil-oy-kun soat:dakika:soniya) qaytaradi.
Bu **NOW()** funksiyasiga teng
Sintaksis:
SELECT CURRENT_TIMESTAMP();

30. MySQL da sana va vaqt bilan ishlash funksiyalari: DATE(), DATEDIFF();

DATE() funksiyasi DATETIME, TIMESTAMP yoki DATE turidagi qiymatdan **faqat sana qismini** (YYYY-MM-DD formatida) ajratib oladi.
DATETIME dan faqat sanani olish
SELECT DATE('2025-05-14 12:34:56');

SELECT DATE(yozilgan_vaqt) AS faqat_sana FROM yangiliklar;
Bu yozilgan_vaqt ustunidagi DATETIME qiymatlardan faqat YYYY-MM-DD qismini chiqaradi.
DATEDIFF() funksiyasi **ikkita sana orasidagi farqni kunlarda** hisoblaydi.
Natija: **butun son** (musbat yoki manfiy).
Ikkita sana orasidagi farq
SELECT DATEDIFF('2025-05-14', '2025-05-01');
Natija: 13
SELECT ism, DATEDIFF(CURDATE(), tugilgan_kuni) AS yosh_kunlarda
FROM talabalar;
Har bir talabaning tug‘ilgan kunidan bugungi kungacha nechta kun o‘tganini qaytaradi.

31. MySQL da sana va vaqt bilan ishlash funksiyalari: DATE_FORMAT(), funksiya parametrlari va ularni misolla bilan tushuntiring;

`DATE_FORMAT()` funksiyasi **sana yoki vaqtni belgilangan format bo‘yicha matn (string)** ko‘rinishida chiqaradi.Asosan **hisobotlar**, **vizual ko‘rinishni yaxshilash**, yoki **faqat kerakli qismlarni ajratib olish** uchun ishlatiladi.

Sintaksis:

DATE_FORMAT(sana_yoki_vaqt, 'format_string')

- sana_yoki_vaqt — DATE, DATETIME, yoki TIMESTAMP qiymati.
- 'format_string' — kerakli chiqish shakli (% bilan boshlanuvchi maxsus belgilar orqali).

Parametr	Ma’nosi
%Y	To‘liq yil (4 raqam)
%y	2 xonali yil
%m	Oy (01–12)

32. MySQL da sana va vaqt bilan ishlash funksiyalari: DAY (), DAYNAME (), DAYOFMONTH, DAYOFWEEK();

DAY() funksiyasi berilgan sananing **kuni (1–31)** qismini qaytaradi.
SELECT DAY('2025-05-14');
Natija: 14
DAYNAME() funksiyasi berilgan sanaga to‘g‘ri keladigan **hafta kunining inglizcha nomini** qaytaradi (masalan: 'Monday', 'Tuesday').
SELECT DAYNAME('2025-05-14');
Natija: 'Wednesday'
DAYOFMONTH() funksiyasi DAY() bilan bir xil: sananing **oydagi kun raqamini (1–31)** qaytaradi.
DAYOFWEEK() funksiyasi berilgan sananing **hafta kunini raqam bilan** qaytaradi:
SELECT DAYOFWEEK('2025-05-14');
Natija: 4

Chorshanba (Wednesday)

33. MySQL MIN() va MAX() funksiyalari;

`MIN()` funksiyasi berilgan ustundagi **eng kichik (minimal)** qiymatni qaytaradi.

```
SELECT MIN(column_name) FROM table_name;
```

Misol:

Agar sizda `students` degan jadval bo‘lsa va unda `age` ustuni bo‘lsa:

```
SELECT MIN(age) AS min_age FROM students;
```

Bu so‘rov `students` jadvalidagi eng yosh talabani qaytaradi.

MAX() funksiyasi berilgan ustundagi **eng katta (maksimal)** qiymatni qaytaradi.

SELECT MAX(column_name) FROM table_name;

SELECT MIN(amount) AS eng_arzon, MAX(amount) AS eng_qimmat FROM orders;

➡️ Bu so‘rov buyurtmalar orasidagi eng arzon va eng qimmat summani beradi.

34. MySQL LIKE operatori. Misollar keltiring;

MySQL'da `LIKE` operatori **qidirilayotgan matnni ma'lum bir andoza (pattern)** bo‘yicha topish uchun ishlatiladi. Bu operator odatda `WHERE` bilan birga ishlatiladi va **qisman mos keladigan qiymatlarni** topadi.Sintaksisi

```
SELECT * FROM table_name
WHERE column_name LIKE 'pattern';
```

Belgi	Ma’nosi
%	0 yoki undan ortiq belgilar o‘rniga ishlatiladi
_	Faqat **1 ta belgi** o‘rniga ishlatiladi

```
SELECT * FROM employees
WHERE name LIKE 'A%';
```

➡️ Masalan: Alice, Alan, Anvar chiqadi.

```
SELECT * FROM employees
WHERE name LIKE '%n';
```

➡️ Masalan: Hasan, Imran

35. MySQL IN operatori. NOT bilan qo’llash. Misollar keltiring;

`IN` operatori **ko‘p qiymatlar orasidan mos keladiganlarini** tanlash uchun ishlatiladi. Bu, bir necha `OR` shartlarini yozishga osonroq alternativ hisoblanadi.

SELECT * FROM table_name
WHERE column_name IN (value1, value2, value3, ...);
SELECT * FROM students
WHERE grade IN (9, 10, 11);
➡️ Bu so‘rov faqat 9-, 10- va 11-sinfdagi o‘quvchilarni chiqaradi.
Aksincha, NOT IN — berilgan qiymatlar **ichida yo‘q** bo‘lganlarini tanlaydi.
SELECT * FROM students
WHERE grade NOT IN (9, 10, 11);
➡️ Bu so‘rov 9-, 10-, va 11-sinfdan boshqa sinflarda o‘qiydiganlarni chiqaradi.

36. MySQL BETWEEN Operatori;

MySQL’da BETWEEN operatori **ikki qiymat oralig‘ida joylashgan ma’lumotlarni** tanlash uchun ishlatiladi. Bu operator **inklyuziv** — ya’ni **chegaradagi qiymatlar ham** hisobga olinadi.
SELECT * FROM table_name
WHERE column_name BETWEEN value1 AND value2;
value1 — pastki chegarasi

value2 — yuqori chegarasi
SELECT * FROM orders
WHERE order_date BETWEEN '2024-01-01' AND '2024-12-31';
➡️ Bu so‘rov 2024-yil davomida olingan buyurtmalarni ko‘rsatadi.

37. MySQL da jadvallarni ulash: INNER JOIN;

MySQL’da `INNER JOIN` ikki yoki undan ortiq **jadvalni umumiy ustun (kalit) orqali bog‘lab**, **faqat mos tushgan satrlarni** (ya’ni ikkala jadvalda ham mavjud bo‘lgan ma’lumotlarni) chiqaradi.

Bu eng ko‘p ishlatiladigan `JOIN` turi hisoblanadi.

SELECT table1.column, table2.column

FROM table1

INNER JOIN table2

ON table1.common_column = table2.common_column;

SELECT s.name, g.group_name

FROM students s

INNER JOIN groups g ON s.group_id = g.id

WHERE g.group_name = 'A guruh';

Faqat A guruhdagi o‘quvchilar.

38. MySQL da jadvallarni ulash: MySQL LEFT JOIN;

MySQL’da LEFT JOIN (yoki LEFT OUTER JOIN) — **chap jadvaldagi barcha satrlarni** va **o‘ng jadvalda mos kelgan satrlarni** qaytaradi.
Agar o‘ng jadvalda mos keladigan satr bo‘lmasa, NULL qiymatlar bilan to‘ldiriladi.
SELECT table1.column, table2.column
FROM table1
LEFT JOIN table2
ON table1.common_column = table2.common_column;
SELECT s.name
FROM students s
LEFT JOIN groups g ON s.group_id = g.id
WHERE g.id IS NULL;
➡️ Faqat guruhga biriktirilmagan o‘quvchilarni ko‘rsatadi.

39. MySQL da jadvallarni ulash: RIGHT JOIN;

MySQL'dagi `RIGHT JOIN` (yoki `RIGHT OUTER JOIN`) — **o'ng jadvaldagi barcha satrlarni**, va chap jadvalda mos kelgan satrlarni chiqaradi. Agar chap jadvalda mos keladigan satr bo'lmasa, u holda `NULL` qiymat bilan to'ldiriladi.

🔸 **Sintaksis:**

```
SELECT table1.column, table2.column
 FROM table1
RIGHT JOIN table2
ON table1.common_column = table2.common_column;

SELECT o.order_id, c.name
FROM orders o
RIGHT JOIN customers c ON o.customer_id = c.id;
```

➡️ Bu so'rov barcha mijozlarni ko'rsatadi, hatto ular hech narsa buyurtma qilmagan bo'lsa ham (order_id = NULL).

JOIN turi	Tavsif
LEFT JOIN	Chap jadvaldagi barcha satrlar + o'ng jadvaldan mos kelganlar
RIGHT JOIN	O'ng jadvaldagi barcha satrlar + chap jadvaldan mos kelganlar
INNER JOIN	Faqat ikkala jadvalda mos tushgan satrlar

40. MySQL da jadvallarni ulash: CROSS JOIN;

MySQL'dagi CROSS JOIN ikki jadval orasidagi **har bir satrni har bir satr bilan juftlaydi** — ya'ni **kartezian ko'paytma (Cartesian Product)** hosil qiladi.
Agar birinchi jadvalda A ta satr, ikkinchisida B ta satr bo'lsa, natijada **A × B** ta satr hosil bo'ladi.
CROSS JOIN odatda:

- **Barcha kombinatsiyalar**ni olish kerak bo'lganda (masalan, mahsulot + variantlar).
- Dinamik hisobotlar yaratishda.
- Test maqsadlarida (kichik jadvallar bilan ishlaganda).

```
SELECT p.product_name, c.color
FROM products p
CROSS JOIN colors c;
```

➡️ Har bir mahsulot uchun har bir rang kombinatsiyasi chiqadi.

41. MySQL UNION operatori;

MySQL'da `UNION` operatori **ikki yoki undan ortiq `SELECT` so‘rovlarining natijalarini birlashtirish** uchun ishlatiladi.
Dublikatlar avtomatik tarzda olib tashlanadi. Agar siz dublikatlarni saqlamoqchi bo‘lsangiz, `UNION ALL` dan foydalaniladi.

SELECT column1, column2, ...

FROM table1

UNION

SELECT column1, column2, ...

FROM table2;

SELECT name, grade FROM students_2024

UNION

SELECT name, grade FROM students_2025;

➡️ Bu so‘rov dublikatlarsiz barcha o‘quvchilar ro‘yxatini beradi.

42. `TRUNCATE TABLE` va `DELETE` orasidagi farq nima?

MySQL’da **TRUNCATE TABLE** va **DELETE** operatorlari **jadvaldagi ma’lumotlarni o‘chirish** uchun ishlatiladi, lekin ular orasida muhim **farqlar mavjud**.

Operator	Tavsifi
DELETE	Jadvaldagi **tanlangan yoki barcha** satrlarni o‘chiradi
TRUNCATE TABLE	Jadvaldagi **barcha satrlarni tozalaydi**, jadvalni **noldan boshlaydi**

DELETE FROM students WHERE grade = 9;
TRUNCATE TABLE students;
-- faqat 9-sinf o‘quvchilarini o‘chiradi

- ✅ TRUNCATE odatda **tezroq**, chunki u butun jadvalni bir operatsiyada tozalaydi.
- ❌ DELETE har bir satrni **birma-bir o‘chiradi**, shuning uchun sekinroq bo‘lishi mumkin.

43. `WHERE` shart operatorining vazifasi nima? Misollar keltiring;

MySQL’da **`WHERE`** operatori **ma'lum shartga mos keladigan satrlarni tanlash** uchun ishlatiladi. Bu operatorni **`SELECT`**, **`UPDATE`**, **`DELETE`** so‘rovlari bilan birga ishlatish mumkin.

`WHERE` operatori orqali siz faqat **ma'lum shartga javob beradigan** satrlarni tanlab, ularni chiqarishingiz, yangilashingiz yoki o‘chirishingiz mumkin.

SELECT name, salary

FROM employees

WHERE salary > 3000;

➡️ Bu so‘rov maoshlari 3000 dan katta bo‘lgan xodimlarni chiqaradi.

44. `ORDER BY` operatori nima uchun ishlatiladi? Misollar keltiring;

MySQL’da **ORDER BY** operatori **natijaviy jadvalni saralash** uchun ishlatiladi. Bu operator yordamida so‘rov natijalarini **yuzaga kelgan tartibda (kamayish yoki o‘sish tartibida)** chiqarish mumkin.
Saralash faqat **SELECT** so‘rovlarida ishlatiladi va natija **o‘zgaradi**.

SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
ASC (default) — o‘sish tartibida (ascending order). Kichikdan kattaga qarab.

DESC — kamayish tartibida (descending order). Kattadan kichikgacha qarab.

SELECT name, salary
FROM employees
ORDER BY salary ASC;

➡️ Bu so‘rov xodimlarni maosh bo‘yicha o‘sish tartibida (kichikdan kattaga) saralaydi.

45. `GROUP BY` operatorining vazifasi nima? Misollar keltiring;

MySQL’da **GROUP BY** operatori **ma'lumotlarni guruhlash** uchun ishlatiladi. Bu operator orqali bir yoki bir nechta ustunlar bo‘yicha **yig‘indilar, o‘rtacha, minimal yoki maksimal qiymatlar** kabi statistik ma'lumotlarni olish mumkin. GROUP BY operatori odatda **agregat funksiyalar** (masalan, COUNT(), SUM(), AVG(), MIN(), MAX()) bilan birgalikda ishlatiladi.
SELECT column1, AGGREGATE_FUNCTION(column2)
FROM table_name GROUP BY column1;
column1 — guruhlashni amalga oshiradigan ustun.
AGGREGATE_FUNCTION(column2) — agregat funksiyasi, masalan COUNT(), SUM(), AVG() va boshqalar.
SELECT department, SUM(salary) AS total_salary
FROM employees
GROUP BY department;
➡️ Bu so‘rov har bir bo‘limdagi jami maoshni hisoblaydi. SUM(salary) agregat funksiyasi orqali bo‘limlar bo‘yicha maoshlar yig‘iladi.

46. `HAVING` va `WHERE` orasidagi farq nima? Misollar keltiring;

MySQL’da **HAVING** va **WHERE** operatorlari **ma'lumotlarni filtrlash** uchun ishlatiladi, lekin ularning ishlash tartibi va maqsadi farq qiladi. **1. WHERE operatori**

- **WHERE** operatori **guruhlashdan oldin** ishlaydi va **guruhlanmagan** satrlarni filtrlash uchun ishlatiladi.
- **WHERE** faqat **oddiy shartlar** (masalan, = , < , >, BETWEEN va boshqalar) bo‘yicha ishlaydi.
- WHERE operatori **agregat funksiyalarni (COUNT, SUM, AVG va h.k.) ishlatish** uchun ishlatilmaydi.
- **HAVING** operatori **guruhlashdan keyin** ishlaydi va **guruhlangan** natijalarni filtrlash uchun ishlatiladi.
- **HAVING** operatori **agregat funksiyalar bilan ishlash** uchun ishlatiladi (masalan, COUNT(), SUM(), AVG() va boshqalar).
- HAVING faqat **GROUP BY** operatori bilan ishlatilganda qo‘llaniladi. **`WHERE`** operatori **guruhlanmagan** natijalarni filtrlash uchun ishlatiladi.

- ⌄ **`HAVING`** operatori esa **guruhlangan** natijalarni filtrlash uchun ishlatiladi va **agregat funksiyalar** bilan ishlaydi.

47. MySQL ANY va ALL operatorlari;

ANY va **ALL** operatorlari MySQL'da **shartlarni solishtirish** uchun ishlatiladi. Ular **subquery** (ichki so‘rovlar) bilan birga ishlatiladi va ma'lum bir qiymatni **bir nechta qiymatlar** bilan taqqoslash imkonini beradi.

ANY operatori: ANY operatori ma'lum bir **qiymatni** subquery orqali olingan natijalarning **barchasi** bilan solishtiradi. Agar subquery natijalaridan **bittasi** ham shartga mos kelsa, umumiy shart **true** bo‘ladi.

*ALL operatori: ALL operatori esa ma'lum bir **qiymatni** subquery orqali olingan natijalarning **hammasi** bilan solishtiradi. Agar subquery natijalari bilan **barchasi** shartga mos kelsa, umumiy shart **true** bo‘ladi.*

```
SELECT column1, column2
FROM table_name
WHERE column1 operator ANY (subquery);
```

operator — masalan, =, >, <, <=, >=, != va hokazo.

subquery — ichki so‘rov (query) bo‘lib, bir yoki bir nechta qiymatlarni qaytaradi.

48. MySQL INSERT INTO SELECT operatorlari;

INSERT INTO SELECT operatori MySQL'da **bitta jadvaldan ma'lumotlarni boshqa jadvalga ko‘chirish** uchun ishlatiladi. Bu operator, ayniqsa, **birinchi jadvaldan ma'lumotlarni tanlab olib, ularni ikkinchi jadvalga kiritish** uchun juda qulaydir. SELECT operatori yordamida ma'lumotlar olinadi, so‘ngra INSERT INTO orqali boshqa jadvalga kiritiladi.

```
INSERT INTO table_name (column1, column2, ...)
SELECT column1, column2, ...
FROM other_table
WHERE condition;
```

table_name — ma'lumotlar kiritiladigan jadval.

column1, column2, ... — kiritiladigan ustunlar (ma'lumotlar).

other_table — ma'lumotlar olinadigan jadval.

condition — tanlab olish uchun qo‘yilgan shart (ixtiyoriy).

49. MySQL CASE operatori;

CASE operatori MySQL’da **shartli ifodalarni** ishlatish uchun qulay vositadir. U orqali ma'lumotlar bazasidagi ma'lum bir shartga asoslanib, **qiymatni o'zgartirish** yoki **turli qiymatlarni tanlash** mumkin. CASE operatori ko'plab **IF** yoki **IF-ELSE** shartlar bilan bir xil maqsadga xizmat qiladi, lekin ko'proq qulay va o'qilishi oson shaklda ishlaydi. SELECT column1,

```
    CASE
        WHEN condition1 THEN result1
        WHEN condition2 THEN result2
        ELSE result3
    END AS new_column
FROM table_name;
```

condition1, condition2 — tekshiriladigan shartlar.

result1, result2, result3 — shartlar bajarilganda qaytariladigan qiymatlar.

new_column — natijalar uchun yangi ustun nomi.

```
SELECT name, salary,
    CASE
        WHEN salary < 3000 THEN 'Low'
            WHEN salary BETWEEN 3000 AND 6000 THEN
'Medium'
        WHEN salary > 6000 THEN 'High'
        ELSE 'Unknown'
    END AS salary_grade
FROM employees
ORDER BY salary_grade;
```

➡️ Bu so‘rov xodimlarni maosh guruhlari bo‘yicha saralaydi.

50. MySQL NULL funksiyalari va izohlar;

MySQL’da NULL — ma'lumotning mavjud emasligini yoki yo'qligini bildiruvchi maxsus qiymatdir. NULL qiymati, noto'g'ri yoki yo'q bo'lgan ma'lumotni ko'rsatadi. Boshqa qiymatlardan farqli o'laroq, NULL bilan ishlashda alohida e'tibor talab qilinadi, chunki u boshqa qiymatlar bilan solishtirilganda hech qanday aniq qiymatga ega emas.

```
SELECT name, salary
FROM employees
WHERE salary IS NULL;
```

➡️ Bu so‘rov salary ustunida qiymati NULL bo‘lgan xodimlarni tanlaydi.

```
SELECT name, COALESCE(salary, 0) AS salary
FROM employees;
```

➡️ Bu so‘rov salary ustuni NULL bo‘lsa, unga 0 qiymati beriladi.

```
SELECT name, NULLIF(salary, 0) AS salary
FROM employees;
```

➡️ Agar salary ustunidagi qiymat 0 bo‘lsa, NULL qaytariladi, aks holda esa salary qiymati saqlanadi.

51.MySQL dagi taqqoslash va mantiqiy operatorlar ;

MySQL – bu keng tarqalgan, kuchli va tezkor ochiq kodli ma'lumotlar bazasini boshqarish tizimi bo‘lib, turli xil so‘rovlar orqali ma'lumotlarga ishlov berish imkoniyatini beradi. Ma'lumotlar ustida mantiqiy tahlil va shartli tanlashni amalga oshirishda **taqqoslash** (comparison) va **mantiqiy** (logical) operatorlarning roli beqiyosdir. Bu operatorlar SELECT, UPDATE, DELETE kabi SQL so‘rovlarida shartlarni aniqlash uchun ishlatiladi.

Taqqoslash operatorlari

Taqqoslash operatorlari ma'lumotlar ustida ikki yoki undan ortiq qiymatni solishtirish uchun qo‘llaniladi. Bu operatorlar yordamida foydalanuvchilar ma'lum qiymatlarga teng, katta, kichik yoki boshqa shartlarga javob beruvchi yozuvlarni tanlab olishlari mumkin.

MySQL'da qo‘llaniladigan asosiy taqqoslash operatorlari quyidagilardan iborat:

- = — Tenglik operatori. Misol: age = 25
- != yoki <> — Teng emaslik operatori. Misol: salary != 1000
- > — Kattaroqlik operatori. Misol: price > 500
- < — Kichikroqlik operatori. Misol: price < 1000
- >= — Kattaroq yoki teng. Misol: rating >= 4.5
- <= — Kichikroq yoki teng. Misol: discount <= 10
- BETWEEN ... AND ... — Belgilangan oraliqda. Misol: age BETWEEN 18 AND 30
- IN (...) — Berilgan qiymatlar ro‘yxatida mavjudligini tekshiradi. Misol: city IN ('Toshkent', 'Samarqand')
- LIKE — Matnli andozalarga mos kelishini aniqlaydi. Misol: name LIKE 'A%'
- IS NULL — NULL qiymat mavjudligini tekshiradi. Misol: email IS NULL
- IS NOT NULL — NULL qiymat mavjud emasligini bildiradi. Misol: email IS NOT NULL

Mantiqiy operatorlar

Mantiqiy operatorlar bir nechta shartlarni birgalikda qo‘llash va ularning mantiqiy bog‘liqligini aniqlash uchun xizmat qiladi. Ular yordamida murakkab shartli ifodalarni yaratish mumkin.

52. MySQL da Viewlar

MySQL’da **View (qarash)** – bu soxta (virtual) jadvallardir. Ular real ma’lumotlarni o‘zida saqlamaydi, balki **asosiy jadvallardagi ma’lumotlarga asoslangan** holda **so‘rov (query)** natijalariga tayangan holda ishlaydi. View’lar orqali murakkab so‘rovlarni soddalashtirish, xavfsizlikni oshirish va ma’lumotlar bazasi tuzilmasini tartibga solish mumkin. View bu – xuddi oddiy jadval kabi ishlatiladi, lekin u aslida SELECT operatoriga asoslangan **saqlangan so‘rov** hisoblanadi. View yaratilganda unda hech qanday real ma’lumot saqlanmaydi – u faqatgina kerakli so‘rovni har safar bajaradi va natijasini qaytaradi.

View yaratishning asosiy sabablari quyidagilardan iborat: birinchidan, **murakkab SQL so‘rovlarini qayta-qayta yozmaslik**, ularni oddiy nom orqali chaqirish imkoniyatidir; ikkinchidan, **xavfsizlik** – foydalanuvchiga faqat kerakli ustunlarni yoki yozuvlarni ko‘rsatish orqali ma’lumotlarning to‘liq emas, balki cheklangan qismi taqdim etiladi; uchinchidan, **loyihaning strukturaviy soddaligi** – ko‘plab jadvallar o‘rniga yagona ko‘rinish taqdim etiladi.

View quyidagi sintaksis orqali yaratiladi:

CREATE VIEW view_nomi AS SELECT_izohi;

Misol uchun, agar bizda students jadvali mavjud bo‘lib, unda barcha talabalarning shaxsiy ma’lumotlari saqlansa, faqat ismi, familiyasi va guruhini o‘z ichiga olgan View quyidagicha yaratiladi:

CREATE VIEW student_short_info AS

SELECT name, surname, group_name

FROM students;

View’lar faqat o‘qish uchun emas, ba’zi hollarda **yangilash, o‘chirish yoki qo‘shish** uchun ham ishlatilishi mumkin. Biroq, bu faqatgina **oddiy View’lar** uchun amal qiladi. Agar View murakkab bo‘lsa (ya’ni, JOIN, GROUP BY, DISTINCT, AGGREGATE FUNCTION kabi operatorlar ishlatilgan bo‘lsa), unda yozish (INSERT, UPDATE, DELETE) operatsiyalari cheklangan bo‘ladi.

Xulosa qilib aytganda, View’lar MySQL ma’lumotlar bazasida **ma’lumotlarga ishlov berish, ularni soddalashtirish, xavfsizlikni ta’minlash va arxitektura yengilligini yaratishda muhim vosita** hisoblanadi. View’lar orqali murakkab so‘rovlar oddiy ko‘rinishda taqdim etiladi va bu esa dasturchilar hamda oxirgi foydalanuvchilar uchun qulaylik yaratadi. Shu bilan birga, View’lar real ma’lumot saqlamagani sababli ularning ishlashi tez va samarali bo‘ladi.

53. MySQL’da saqlanadigan proseduralar

MySQL’da **saqlanadigan proseduralar (stored procedures)** – bu ma’lumotlar bazasida bir marta yaratiladigan va keyinchalik takror-takror chaqiriladigan SQL buyruqlar majmuasidir. Ular odatda murakkab yoki takrorlanuvchi amallarni avtomatlashtirish, kodni modullashtirish va ma’lumotlar bazasi ishlashini optimallashtirish maqsadida qo‘llaniladi. Saqlanadigan proseduralar yordamida foydalanuvchi bir necha SQL operatorlarini yagona blokda jamlab, ularni parametrlar bilan boshqarishi mumkin. Bunday yondashuv dastur kodini soddalashtiradi, xatoliklarni kamaytiradi va ishlash samaradorligini oshiradi. Prosedura server darajasida saqlanadi va chaqirilganda avtomatik ravishda bajariladi.

MySQL’da prosedura quyidagi sintaksis orqali yaratiladi:

```
DELIMITER //

CREATE PROCEDURE prosedura_nomi([parametrlar])

BEGIN

  -- SQL buyruqlar

END // DELIMITER ;
```

Bu yerda DELIMITER buyrug‘i MySQL’ga kod blokining qayerda tugashini ko‘rsatadi. Standart ; ajratgich o‘rniga // ishlatiladi, chunki prosedura ichida ko‘plab ; belgilar mavjud bo‘ladi.

Prosedura parametrlar bilan ishlaydi va bu parametrlar quyidagi turlarga bo‘linadi:

1. **IN** – kirish parametri; proseduraga qiymat yuboriladi.
2. **OUT** – chiqish parametri; prosedura bajarilgandan so‘ng qiymat qaytaradi.
3. **INOUT** – kirish va chiqish; qiymat yuboriladi va o‘zgartirilgan holda qaytadi.

Quyidagi misolda oddiy prosedura berilgan bo‘lib, u foydalanuvchining to‘liq ismini chiqaradi:

```
DELIMITER //

CREATE PROCEDURE getFullName(IN userId INT)

BEGIN

  SELECT CONCAT(first_name, ' ', last_name) AS full_name

  FROM users

  WHERE id = userId;    END //    DELIMITER ;
```

54. MySQL’da tranzaksiyalar

MySQL’da **tranzaksiya (transaction)** – bu ma’lumotlar bazasida bajarilishi kerak bo‘lgan bir yoki bir nechta SQL amallarining mantiqiy guruhidir. Tranzaksiyalar yordamida ma’lumotlar ustida bir nechta amallarni ketma-ket bajarish va ularning **yagona operatsiya sifatida** amalga oshirilishini ta’minlash mumkin. Tranzaksiya to‘liq bajarilishi yoki umuman bajarilmasligi kerak. Bu yondashuv **ACID** tamoyillariga asoslanadi: *Atomicity (atomiklik), Consistency (izchillik), Isolation (izolyatsiya), Durability (barqarorlik)*. MySQL’da tranzaksiyalar asosan **InnoDB** turidagi jadvallar bilan qo‘llab-quvvatlanadi.

Atomiklik – bu tranzaksiyadagi barcha operatsiyalar yoki to‘liq bajariladi, yoki umuman bajarilmaydi. Agar biror bosqichda xatolik yuz bersa, butun tranzaksiya bekor qilinadi. Izchillik – tranzaksiya bajarilishidan oldin va keyin ma’lumotlar bazasi izchil holatda bo‘lishi kerak. Izolyatsiya – har bir tranzaksiya mustaqil bajariladi va boshqa tranzaksiyalarga ta’sir qilmaydi. Barqarorlik – tranzaksiya bajarilib, natijalar saqlangandan so‘ng ular tizim nosozligiga qaramay saqlanib qoladi.

MySQL’da tranzaksiya quyidagi buyruqlar orqali boshqariladi:

- START TRANSACTION yoki BEGIN – tranzaksiyani boshlash
- COMMIT – tranzaksiyani yakunlash va o‘zgarishlarni saqlash
- ROLLBACK – xatolik yuz bersa tranzaksiyani bekor qilish va ma’lumotlarni dastlabki holatga qaytarish

Quyidagi misolda tranzaksiya orqali bank hisoblaridan pul o‘tkazish jarayoni ifodalangan:

```
START TRANSACTION;

UPDATE accounts SET balance = balance - 500 WHERE account_id = 1;

UPDATE accounts SET balance = balance + 500 WHERE account_id = 2;

COMMIT;
```

Tranzaksiyalar ma’lumotlar bazasining ishonchliligini ta’minlashda, ayniqsa ko‘p foydalanuvchili va real vaqtli tizimlarda muhim rol o‘ynaydi. Ular yordamida ma’lumotlar ustida bajariladigan amallar to‘liq nazorat ostida bo‘lib, noto‘g‘ri yoki to‘liq bajarilmagan amallar tufayli yuzaga keladigan xatoliklar oldi olinadi.

55. MySQL ga ulanish uchun C# dasturlash tilida Dbconnection deb nomlanuvchi sinfi tuzib bering;

```
using System;
using MySql.Data.MySqlClient;

namespace MahsulotMagazini
{
    public class DbConnection
    {
        // Ulanish satri (connection string) – o'z bazangizga moslab o'zgartiring
        private string connectionString =
"Server=localhost;Database=mahsulotdb;Uid=root;Pwd=your_password;";

        // MySQL ulanish obyekti
        private MySqlConnection connection;

        // Konstruktor – ulanish obyekti yaratiladi
        public DbConnection()
        {
            connection = new MySqlConnection(connectionString);
        }

        // Ulanishni ochish
        public void OpenConnection()
        {
            try
            {
                if (connection.State == System.Data.ConnectionState.Closed)
                {
                    connection.Open();
                }
```

```
        }
        catch (MySqlException ex)
        {
            Console.WriteLine("Ulanishda xatolik: " + ex.Message);
        }
    }
    // Ulanishni yopish
    public void CloseConnection()
    {
        try
        {
            if (connection.State == System.Data.ConnectionState.Open)
            {
                connection.Close();
            }
        }
        catch (MySqlException ex)
        {
            Console.WriteLine("Yopishda xatolik: " + ex.Message);
        }
    }
    // Ulanish obyektini olish (boshqa formalar foydalanishi uchun)
    public MySqlConnection GetConnection()
    {
        return connection;
    }
}}
```

56. C# dasturlash tilida Users jadvalini bo’g’lanib, foydalanuvchini avtorizatsiydan o’tish kodini yozing;

1. Users jadvali tuzilmasi (MySQL):

```
CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(100) NOT NULL
);
```

2. LoginForm uchun C# kod (autentifikatsiya logikasi):

```
using System;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace MahsulotMagazini
{
    public partial class LoginForm : Form {
        DbConnection db = new DbConnection();
        public LoginForm()     {
            InitializeComponent(); }
        private void btnLogin_Click(object sender, EventArgs e)  {
            string username = txtUsername.Text.Trim();
            string password = txtPassword.Text.Trim();
            if (username == "" || password == "") {
                MessageBox.Show("Iltimos, login va parolni to'ldiring.");
                return; }
            try{
                db.OpenConnection();
                string query = "SELECT * FROM Users WHERE username =
@username AND password = @password";
                MySqlCommand cmd = new MySqlCommand(query,
db.GetConnection());
```

```
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", password);

            MySqlDataReader reader = cmd.ExecuteReader();

            if (reader.HasRows)
            {
                MessageBox.Show("Tizimga muvaffaqiyatli kirdingiz!");
                reader.Close();
                db.CloseConnection();

                // Asosiy formani ochish
                MainForm mainForm = new MainForm();
                mainForm.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Login yoki parol noto‘g‘ri.");
                reader.Close();
                db.CloseConnection();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Xatolik: " + ex.Message);
        }  }   }}
```

57. DataGridView komponentida MySQL ma’lumotlarini tanlab olish;

1. MySQL jadvali namunasi:

```
CREATE TABLE Products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100),
    price DECIMAL(10,2),
    quantity INT);
```

2. C# kod: MySQL ma’lumotlarini DataGridView ga yuklash:

```
using System;
using System.Data;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
namespace MahsulotMagazini{
    public partial class ProductForm : Form  {
        DbConnection db = new DbConnection(); // Oldindan yozilgan MySQL ulanish sinfi
        public ProductForm()  {
            InitializeComponent()  }
        private void btnLoad_Click(object sender, EventArgs e){     try {
                db.OpenConnection(); // Ulanishni ochamiz
                string query = "SELECT * FROM Products"; // Tanlab olish so‘rovi
                MySqlCommand cmd = new MySqlCommand(query, db.GetConnection());
                MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
                DataTable table = new DataTable();
                adapter.Fill(table); // Ma’lumotlarni DataTable ga yuklaymiz
                dataGridView1.DataSource = table; // DataGridView ga biriktiramiz
                db.CloseConnection(); // Ulanishni yopamiz   }  catch (Exception ex) {
 MessageBox.Show("Xatolik yuz berdi: " + ex.Message);}}}}
```

58. Formada filtrlash amallarini bajarish;

1. Jadval va Ma'lumotlar:

Products jadvali:

```
CREATE TABLE Products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100),
    price DECIMAL(10, 2),
    quantity INT
);
```

2. C# kod: Filtrlash amallari

```
using System;
using System.Data;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
namespace MahsulotMagazini
{
    public partial class ProductForm : Form
    {
        DbConnection db = new DbConnection(); // Oldindan yozilgan MySQL ulanish sinfi
        public ProductForm()
        {
            InitializeComponent();
        }
        // Filtrlash tugmasi bosilganida
        private void btnFilter_Click(object sender, EventArgs e)
        {
            string filterValue = txtSearch.Text.Trim();
            if (string.IsNullOrEmpty(filterValue))
            {
                MessageBox.Show("Iltimos, filtrlash uchun qiymat kiriting.");
```

```
            return;
        }
        try
        {
            db.OpenConnection(); // MySQL ulanishini ochish
            // Filtrlash so‘rovini tuzish
            string query = "SELECT * FROM Products WHERE product_name
LIKE @filterValue OR price LIKE @filterValue";
            MySqlCommand cmd = new MySqlCommand(query,
db.GetConnection());
            cmd.Parameters.AddWithValue("@filterValue", "%" + filterValue +
"%");
            MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
            DataTable table = new DataTable();
            adapter.Fill(table);
            // DataGridView ga ma'lumotlarni yuklash
            dataGridView1.DataSource = table;
            db.CloseConnection(); // Ulanishni yopish
        }
        catch (Exception ex)
        {
            MessageBox.Show("Xatolik yuz berdi: " + ex.Message);
        }
    }
    // Form yuklanganda barcha ma'lumotlarni ko'rsatish
    private void ProductForm_Load(object sender, EventArgs e)
    {
        LoadAllData();
    }
    // Barcha ma'lumotlarni yuklash
```

```
        private void LoadAllData()
        {
            try
            {
                db.OpenConnection(); // MySQL ulanishini ochish
                string query = "SELECT * FROM Products"; // Barcha ma'lumotlarni
olish
                MySqlCommand cmd = new MySqlCommand(query,
db.GetConnection());
                MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
                DataTable table = new DataTable();
                adapter.Fill(table);
                dataGridView1.DataSource = table; // DataGridView ga ma'lumotlarni
yuklash
                db.CloseConnection(); // Ulanishni yopish
            }
            catch (Exception ex)
            {
                MessageBox.Show("Xatolik yuz berdi: " + ex.Message);}}}}
```

59. Formada like orqali filtrlash amallarini bajarish;

"Formada LIKE operatori orqali filtrlash amallarini bajarish" deganda, foydalanuvchi kiritgan matn asosida ma’lumotlarni qidirish (filtrlash) tushuniladi. Bunda SQL tilidagi LIKE operatori yordamida belgilangan shartga mos keladigan yozuvlar tanlab olinadi.

LIKE operatori matnli ustunlar ustida ishlaydi va WHERE sharti bilan birga qo‘llanadi.

Sintaksis:

SELECT * FROM jadval_nomi

WHERE ustun_nomi LIKE 'shart';

1. Ism “Ali” bilan boshlanadigan mijozlarni chiqarish:

SELECT * FROM Customers

WHERE Name LIKE 'Ali%';

Bu so‘rov Customers jadvalidan ismi Ali, Alisher, Aliya kabi boshlanuvchi yozuvlarni chiqaradi.

2. Mahsulot nomi "kola" bilan tugaydigan mahsulotlarni olish:

SELECT * FROM Products

WHERE ProductName LIKE '%kola';

Natijada Coca Kola, Pepsi Kola kabi mahsulotlar chiqadi.

3. Ichida "sam" bo‘lgan mahsulotlarni olish:

SELECT * FROM Products

WHERE ProductName LIKE '%sam%';

Bu so‘rov nomining ichida sam bo‘lgan mahsulotlarni qaytaradi (masalan: Samsung, Assam).

60. Combobox elimentida MySQL ma’lumotlarini tanlab olish;

```
using MySql.Data.MySqlClient;
string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
MySqlConnection connection = new MySqlConnection(connectionString);
try
{
    connection.Open();
    string query = "SELECT kategoriya_nomi FROM kategoriya";
    MySqlCommand cmd = new MySqlCommand(query, connection);
    MySqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        comboBox1.Items.Add(reader["kategoriya_nomi"].ToString());
    }
    reader.Close();
    connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Xatolik: " + ex.Message);
}
```

61. Combobox elimenti tanlaganda DataGridViewda filtrlashni qo’llash;

1. ComboBox’ni to‘ldirish (kategoriya_nomi bilan)

```
private void LoadCategories()
{
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        string query = "SELECT kategoriya_nomi FROM kategoriya";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            comboBox1.Items.Add(reader["kategoriya_nomi"].ToString());
        }
        reader.Close();
    }
}
```

Bu metodni Form1_Load da chaqiring.

---

2. DataGridViewni to‘ldirish (filtr bilan)

```
private void FilterProductsByCategory(string kategoriya)
{
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        string query = "SELECT * FROM mahsulotlar WHERE kategoriya_nomi =
@kategoriya";
```

```
        MySqlCommand cmd = new MySqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@kategoriya", kategoriya);

        MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);
        DataTable table = new DataTable();
        adapter.Fill(table);

        dataGridView1.DataSource = table;
    }
}
```

---

3. ComboBox tanlanganda ishga tushadigan kod

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectedCategory = comboBox1.SelectedItem.ToString();
    FilterProductsByCategory(selectedCategory);
}
```

62.statusStrip componentida bazada mavjud bo’lgan yozuvlar sonini olib kelish;

StatusStrip komponentida bazada mavjud bo‘lgan yozuvlar sonini olish uchun SQL so‘rovi yordamida jadvaldagi yozuvlar sonini hisoblash talab qilinadi. Bu maqsadda SELECT COUNT(*) FROM jadval_nomi ko‘rinishidagi SQL so‘rovi ishlatiladi. COUNT(*) funksiyasi jadvaldagi barcha yozuvlar sonini hisoblaydi. Masalan, agar sizda mahsulotlar nomli jadval bo‘lsa, so‘rov quyidagicha yoziladi:
SELECT COUNT(*) FROM mahsulotlar
Bu so‘rov mahsulotlar jadvalidagi mavjud barcha qatormlar sonini hisoblab qaytaradi. Ushbu qiymatni C# dasturida olish uchun ExecuteScalar() metodi ishlatiladi. Bu metod faqat bitta qiymatni qaytaradi va bu bizga aynan kerakli holat, chunki COUNT(*) faqat bitta raqam beradi. Olingan qiymat ToolStripStatusLabel komponentiga yoziladi. Masalan:

```
string query = "SELECT COUNT(*) FROM mahsulotlar";

MySqlCommand cmd = new MySqlCommand(query, connection);

int count = Convert.ToInt32(cmd.ExecuteScalar());

toolStripStatusLabel1.Text = "Yozuvlar soni: " + count.ToString();
```

Bu kodda query o‘zgaruvchisi SQL so‘rovni o‘z ichiga oladi, cmd.ExecuteScalar() orqali yozuvlar soni olinadi va toolStripStatusLabel1 ustuniga chiqariladi. Shu tariqa, SQL so‘rovi orqali yozuvlar soni hisoblanadi va StatusStripda ko‘rsatiladi.

63.DataGridView komponentida tanlanagan yozuvlarni tuzatish rejimi uchun komponentlarga joylashtirish;

DataGridViewning CellClick yoki SelectionChanged hodisasida tanlangan qatordagi ma’lumotlarni TextBoxlarga yozamiz:

```
private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)

{
   if (e.RowIndex >= 0)
   {
      DataGridViewRow row = dataGridView1.Rows[e.RowIndex];
      txtNomi.Text = row.Cells["nomi"].Value.ToString();
      txtNarxi.Text = row.Cells["narxi"].Value.ToString();
      txtSoni.Text = row.Cells["soni"].Value.ToString();
   }
}
```

2. Yangilash tugmasi orqali ma’lumotni bazada saqlash

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        // Faraz qilaylik: mahsulotlar jadvalida id mavjud
        int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells["id"].Value);
        string query = "UPDATE mahsulotlar SET nomi=@nomi, narxi=@narxi,
soni=@soni WHERE id=@id";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@nomi", txtNomi.Text);
        cmd.Parameters.AddWithValue("@narxi", txtNarxi.Text);
        cmd.Parameters.AddWithValue("@soni", txtSoni.Text);
        cmd.Parameters.AddWithValue("@id", id);
        cmd.ExecuteNonQuery();
    }
    MessageBox.Show("Yozuv yangilandi!");
    LoadData(); // DGVni yangilash funksiyasi
}
```

---

64. Ma’lumotlarni qo’shish uchun formada tayyorlash;

Yozuv qo‘shishda quyidagi so‘rov ishlatiladi:

INSERT INTO mahsulotlar (nomi, narxi, soni) VALUES ('Cola', 5000, 10)

Bu so‘rov C# da parametrlar bilan yoziladi, xakerlik (SQL Injection) xavfini kamaytirish uchun.

1. Qo‘shish tugmasi kodi (btnAdd_Click)

```
private void btnAdd_Click(object sender, EventArgs e)
{
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        string query = "INSERT INTO mahsulotlar (nomi, narxi, soni) VALUES
(@nomi, @narxi, @soni)";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@nomi", txtNomi.Text);
        cmd.Parameters.AddWithValue("@narxi",
Convert.ToDecimal(txtNarxi.Text));
        cmd.Parameters.AddWithValue("@soni", Convert.ToInt32(txtSoni.Text));
        cmd.ExecuteNonQuery()  }
    MessageBox.Show("Yangi mahsulot qo‘shildi!");
    this.Close(); // forma yopiladi}
```

2. Qo‘shish formasini asosiy formadan chaqirish:

Agar bu formani MainForm dan tugma orqali ochmoqchi bo‘lsangiz:

```
private void btnAddNew_Click(object sender, EventArgs e)
{
    AddForm form = new AddForm();
    form.ShowDialog();
    LoadData(); // ma'lumotlarni qayta yuklash}
```

65. Bazaga yangi yozuvni qo’shish amalini C# dasturlash tili orqali amalga oshiring;

```
using MySql.Data.MySqlClient;
string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
```

1. INSERT SQL so‘rovi va C# kodi:

```
private void YozuvQoshish()
{
    // Ma'lumotlar (odatda formadagi TextBox orqali olinadi)
    string nomi = txtNomi.Text;
    decimal narxi = Convert.ToDecimal(txtNarxi.Text);
    int soni = Convert.ToInt32(txtSoni.Text);
    // Ulanish
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        // SQL so‘rovi
        string query = "INSERT INTO mahsulotlar (nomi, narxi, soni) VALUES
(@nomi, @narxi, @soni)";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        // Parametrlar biriktirish
        cmd.Parameters.AddWithValue("@nomi", nomi);
        cmd.Parameters.AddWithValue("@narxi", narxi);
        cmd.Parameters.AddWithValue("@soni", soni);

        // So‘rovni bajarish
        cmd.ExecuteNonQuery();
        MessageBox.Show("Yozuv muvaffaqiyatli qo‘shildi!");}}
```

66. Bazada mavjud yozuvni o’zgartish amalini C# dasturlash tili orqali amalga oshiring;

1. UPDATE SQL so‘rovi va kod:

```
private void YozuvniYangilash()
{
    // Tanlangan yozuvning ID qiymatini olish
    int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells["id"].Value);
    // O'zgartirilgan qiymatlar
    string nomi = txtNomi.Text;
    decimal narxi = Convert.ToDecimal(txtNarxi.Text);
    int soni = Convert.ToInt32(txtSoni.Text);
    // MySQL ulanish satri
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        // SQL so‘rovi
        string query = "UPDATE mahsulotlar SET nomi=@nomi, narxi=@narxi,
soni=@soni WHERE id=@id";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        // Parametrlar qo‘shish
        cmd.Parameters.AddWithValue("@nomi", nomi);
        cmd.Parameters.AddWithValue("@narxi", narxi);
        cmd.Parameters.AddWithValue("@soni", soni);
        cmd.Parameters.AddWithValue("@id", id);
        // So‘rovni bajarish
        cmd.ExecuteNonQuery();
        MessageBox.Show("Yozuv yangilandi!");
    }
}
```

2. Button click voqeasida chaqirish:

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    YozuvniYangilash();
    LoadData(); // DataGridViewni qayta yuklash funksiyasi
}
```

3. Tanlangan yozuvni TextBoxlarga chiqarish (masalan CellClick orqali):

```
private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];
        txtNomi.Text = row.Cells["nomi"].Value.ToString();
        txtNarxi.Text = row.Cells["narxi"].Value.ToString();
        txtSoni.Text = row.Cells["soni"].Value.ToString();
    }
}
```

67. Bazada mavjud yozuvni o’chirish amalini C# dasturlash tili orqali amalga oshiring;

1. SQL DELETE so‘rovi va C# kodi:

```
private void YozuvniOchirish()
```

```
{
    // Tanlangan yozuvning ID qiymatini olish
    int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells["id"].Value);
    // MySQL ulanish satri
    string connectionString =
"server=localhost;database=mahsulotlar;uid=root;pwd=;";
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        // DELETE SQL so‘rovi
        string query = "DELETE FROM mahsulotlar WHERE id=@id";
        MySqlCommand cmd = new MySqlCommand(query, conn);
        // Parametrni qo‘shish
        cmd.Parameters.AddWithValue("@id", id);
        // So‘rovni bajarish
        cmd.ExecuteNonQuery();
        MessageBox.Show("Yozuv o‘chirildi!");
    }
}
```

2. Button click voqeasida chaqirish:

```
private void btnDelete_Click(object sender, EventArgs e)
{
    // Foydalanuvchidan tasdiq olish
    DialogResult result = MessageBox.Show("Siz haqiqatan ham yozuvni
o'chirishni xohlaysizmi?", "Tasdiqlash", MessageBoxButtons.YesNo);
    if (result == DialogResult.Yes)
    {
        YozuvniOchirish();
```

```
        LoadData(); // DataGridViewni qayta yuklash funksiyasi
    }
}
```

3. Tanlangan yozuvni o‘chirishdan oldin tasdiqlash:

MessageBox orqali foydalanuvchidan tasdiq olish:

```
DialogResult result = MessageBox.Show("Yozuvni o‘chirishni xohlaysizmi?", "Tasdiqlash", MessageBoxButtons.YesNo);
if (result == DialogResult.Yes)
{
    // Yozuvni o‘chirish
}
```

68. Foydalanuvchilar jadvalidagi userid ning maksimal qiymatini topishni C# dasturlash tili orqali amalga oshiring;

1. SQL so‘rovini yozish:

Bunda MAX SQL funksiyasidan foydalanamiz:

```
SELECT MAX(userid) FROM foydalanuvchilar;
```

Bu so‘rov foydalanuvchilar jadvalidagi userid ustunidagi eng katta qiymatni qaytaradi.

2**. C# kodi orqali maksimal useridni olish:**

```
private void GetMaxUserId()
{
   // MySQL ulanish satri
   string connectionString =
"server=localhost;database=foydalanuvchilar;uid=root;pwd=;";
   using (MySqlConnection conn = new MySqlConnection(connectionString)  {
      try{
         conn.Open();
         // SQL so‘rovi
         string query = "SELECT MAX(userid) FROM foydalanuvchilar";
         MySqlCommand cmd = new MySqlCommand(query, conn);
         // So‘rovni bajarish va natijani olish
         object result = cmd.ExecuteScalar();
         // Natijani tekshirish
         if (result != DBNull.Value) {
            int maxUserId = Convert.ToInt32(result);
            MessageBox.Show("Eng katta userid: " + maxUserId)   }   else {
            MessageBox.Show("Foydalanuvchilar mavjud emas.");
         } }
      catch (Exception ex){
         MessageBox.Show("Xato yuz berdi: " + ex.Message);}}}
```

69. Bazaga bool tipini yozishda foydalanuvchi oynasiga chekbox elimenti orqali amalga oshirish jarayoni kodini yozib bering (chekboxni statusini bazaga yozish);

C# kodi:

1. **CheckBoxdan qiymat olish va bazaga yozish**:

```
    private void SaveToDatabase()
{
   string connectionString =
"server=localhost;database=foydalanuvchilar;uid=root;pwd=;";
   bool isActive = chkActive.Checked;
   string username = txtUsername.Text;
   // MySQL ulanish
   using (MySqlConnection conn = new MySqlConnection(connectionString))
   {try
      {
         conn.Open();
         string query = "INSERT INTO foydalanuvchilar (username, is_active)
VALUES (@username, @is_active)";
         MySqlCommand cmd = new MySqlCommand(query, conn);
         cmd.Parameters.AddWithValue("@username", username);
         cmd.Parameters.AddWithValue("@is_active", isActive ? 1 : 0);  //
CheckBox: True => 1, False => 0
         cmd.ExecuteNonQuery();
         MessageBox.Show("Ma'lumot bazaga yozildi!");
      }
      catch (Exception ex)
      {
         MessageBox.Show("Xato yuz berdi: " + ex.Message);}}}
```

70. NOTNULL qiymatlani bazaga yozmaslik uchun formada qanday cheklovlar o’rnatish kerak?

MySQLda NOT NULL cheklovini ustun uchun quyidagicha belgilaysiz:

```
CREATE TABLE foydalanuvchilar (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
```

```
    email VARCHAR(100) NOT NULL,
    is_active TINYINT(1) NOT NULL
);
```

Bu yerda username, email, va is_active ustunlari NOT NULL bo‘lib, ularni bo‘sh qiymat bilan bazaga yozish mumkin emas.

2. C# forma uchun cheklovlarni o‘rnatish:

Agar foydalanuvchi NOT NULL maydonlarini to‘ldirmagan bo‘lsa, ularni bazaga yozishdan oldin xatolikni ko‘rsatishimiz kerak.

Misol uchun, foydalanuvchi username va email ni to‘ldirishini talab qilamiz.

3. Cheklovlarni o‘rnatish va xatoliklarni ko‘rsatish:

```
private bool ValidateForm()
{
    // Formadagi ma'lumotlarni tekshirish
    if (string.IsNullOrEmpty(txtUsername.Text))
    {
        MessageBox.Show("Username maydoni bo'sh bo'lishi mumkin emas.");
        return false; // Username bo'sh bo'lsa, so'rovni yuborishdan to'xtatadi
    }
    if (string.IsNullOrEmpty(txtEmail.Text))
    {
        MessageBox.Show("Email maydoni bo'sh bo'lishi mumkin emas.");
        return false; // Email bo'sh bo'lsa, so'rovni yuborishdan to'xtatadi
    }
    // Boshqa cheklovlar (agar kerak bo'lsa)
    // Masalan, emailni formatini tekshirish
    if (!txtEmail.Text.Contains("@"))
    {
```

```
        MessageBox.Show("Email manzili noto'g'ri formatda.");
        return false;
    }
    return true; // Hamma maydonlar to'ldirilgan bo'lsa, true qaytaradi
}
```

4. Ma'lumotlarni bazaga yozishdan oldin ValidateForm() funksiyasini chaqirish:

```
private void btnSave_Click(object sender, EventArgs e)
{
    // Formadagi ma'lumotlarni tekshirish
    if (ValidateForm())
    {
        // Agar form valid bo'lsa, ma'lumotni bazaga yozish
        SaveToDatabase();
    }
}
```

71. MySQL MBBT da Universitet bazasi ichida talabalar jadvalini yaratish uchun SQL so’rovini yozib bering;

1. Universitet bazasini yaratish:

Avvalo, agar **Universitet** bazasi hali yaratilmagan bo'lsa, uni yaratish kerak:

```
CREATE DATABASE IF NOT EXISTS universitet;
USE universitet;
```

2. Talabalar jadvalini yaratish:

Talabalar jadvalini yaratishda quyidagi ma'lumotlarni kiritamiz:

- id: Talabaning unikal identifikatori (Primary Key).
- ism: Talabaning ismi.
- familiya: Talabaning familiyasi.
- tugilgan_yil: Talabaning tug‘ilgan yili.
- email: Talabaning elektron pochta manzili.
- telefon: Talabaning telefon raqami.
- fakultet: Talaba o‘qiyotgan fakultet nomi.
- kurs: Talaba o‘qiyotgan kurs raqami.
- is_active: Talabaning faol yoki faol emasligi (boolean qiymat).

```
CREATE TABLE IF NOT EXISTS talabalar (
    id INT AUTO_INCREMENT PRIMARY KEY,        -- Talabaning unikal identifikatori
    ism VARCHAR(50) NOT NULL,                 -- Talabaning ismi
    familiya VARCHAR(50) NOT NULL,            -- Talabaning familiyasi
    tugilgan_yil INT NOT NULL,                -- Talabaning tug‘ilgan yili
    email VARCHAR(100) NOT NULL,              -- Talabaning elektron pochta manzili
    telefon VARCHAR(15) NOT NULL,             -- Talabaning telefon raqami
    fakultet VARCHAR(100) NOT NULL,           -- Talaba o‘qiyotgan fakultet nomi
    kurs INT NOT NULL,                        -- Talaba o‘qiyotgan kurs
    is_active TINYINT(1) DEFAULT 1            -- Talabaning faol yoki faol emasligi (1 = Faol, 0 = Faol emas)
);
```

3. Tavsif:

- **id**: Talabaning unikal identifikatori. AUTO_INCREMENT yordamida har safar yangi talabani qo‘shganingizda avtomatik ravishda qiymat ortadi.
- **ism** va **familiya**: Talabaning ismi va familiyasi. **NOT NULL** qo‘yilgan, shuning uchun bu maydonlar bo‘sh bo‘lishi mumkin emas.

- **tugilgan_yil**: Talabaning tug‘ilgan yili.
- **email** va **telefon**: Talabaning aloqa ma'lumotlari.
- **fakultet**: Talabaning o‘qiyotgan fakulteti nomi.
- **kurs**: Talaba o‘qiyotgan kurs raqami.
- **is_active**: Talabaning faol yoki faol emasligi. TINYINT(1) qiymati, ya'ni 0 — faol emas, 1 — faol.

4. Jadvalni yaratish va ma'lumot qo‘shish:

Jadval yaratilib bo‘lgach, unga quyidagi tarzda ma'lumotlar qo‘shish mumkin:

INSERT INTO talabalar (ism, familiya, tugilgan_yil, email, telefon, fakultet, kurs, is_active)

VALUES

('Ali', 'Rahmonov', 2000, 'ali.rahmonov@univ.com', '998901234567', 'Informatika', 1, 1),

('Nodira', 'Xamidova', 1999, 'nodira.xamidova@univ.com', '998905678901', 'Matematika', 2, 1),

('Jahongir', 'Tursunov', 2001, 'jahongir.tursunov@univ.com', '998907654321', 'Fizika', 1, 0);

72. MySQL MBBT da Universitet bazasi ichida talabalar jadvalini tuzatish uchun SQL so’rovini yozib bering;

Misol 1: Agar talabalar jadvaliga **address** (manzil) nomli yangi ustun qo‘shish kerak bo‘lsa, quyidagi so‘rovni ishlatamiz:

ALTER TABLE talabalar

ADD COLUMN address VARCHAR(255) NOT NULL;

Misol 2: Ustun qiymatini yangilash

Agar talabalar jadvalidagi mavjud talabani yangilamoqchi bo‘lsak, masalan, id = 1 bo‘lgan talabaning **kurs** raqamini o‘zgartirish:

UPDATE talabalar

SET kurs = 2

WHERE id = 1;

Bu so‘rov **id = 1** bo‘lgan talabaning **kurs** qiymatini **2** ga o‘zgartiradi.

73. MySQL MBBT da Universitet bazasi ichida talabalar jadvaliga yangi ma’lumot qo’shish uchun SQL so’rovini yozib bering;

1. INSERT INTO so'rovi:

Yangi ma'lumot qo'shish uchun **INSERT INTO** SQL so'rovini ishlatamiz. Quyida **Talabalar** jadvaliga yangi talabani qo'shish uchun so'rov ko'rsatilgan.

Misol: Yangi talabani qo'shish

INSERT INTO talabalar (ism, familiya, tugilgan_yil, email, telefon, fakultet, kurs, is_active)

VALUES ('Ali', 'Rahmonov', 2000, 'ali.rahmonov@univ.com', '998901234567', 'Informatika', 1, 1);

---

2. So'rovni tushuntirish:

INSERT INTO talabalar: Bu so'rov **talabalar** jadvaliga yangi ma'lumot qo'shadi.

(ism, familiya, tugilgan_yil, email, telefon, fakultet, kurs, is_active): Bu joyda jadval ustunlari nomlarini belgilaymiz, ya'ni qanday ustunlarga ma'lumot qo'shayotganimizni ko'rsatamiz.

VALUES ('Ali', 'Rahmonov', 2000, 'ali.rahmonov@univ.com', '998901234567', 'Informatika', 1, 1);: Bu qismda qo'shmoqchi bo'lgan talabaning ma'lumotlarini kiritamiz.

Ali: Talabaning ismi.

Rahmonov: Talabaning familiyasi.

2000: Talabaning tug‘ilgan yili.

ali.rahmonov@univ.com: Talabaning email manzili.

998901234567: Talabaning telefon raqami.

Informatika: Talaba o‘qiyotgan fakultet.

1: Talaba 1-kursda o‘qiydi.

1: Talaba faol (1 - faol, 0 - faol emas).

3. Boshqa misollar:

Misol 2: Ikki talaba qo'shish

INSERT INTO talabalar (ism, familiya, tugilgan_yil, email, telefon, fakultet, kurs, is_active)

VALUES

('Nodira', 'Xamidova', 1999, 'nodira.xamidova@univ.com', '998905678901', 'Matematika', 2, 1),

('Jahongir', 'Tursunov', 2001, 'jahongir.tursunov@univ.com', '998907654321', 'Fizika', 1, 0);

74. MySQL MBBT da Universitet bazasi ichida talabalar jadvalidagi ma’lumotlarni o’chirish uchun SQL so’rovini yozib bering;

1. DELETE FROM so'rovi:

Ma'lumotlarni o‘chirish uchun **DELETE FROM** SQL so'rovini ishlatamiz. So'rovda kerakli talabani o‘chirish uchun **WHERE** shartini qo‘shish muhim, aks holda barcha ma'lumotlar o‘chib ketishi mumkin.

Misol 1: id bo‘yicha talabani o‘chirish

Agar **id = 1** bo‘lgan talaba ma'lumotlarini o‘chirmoqchi bo‘lsak:

```
DELETE FROM talabalar
WHERE id = 1;
```

Bu so‘rov **id = 1** bo‘lgan talaba ma'lumotlarini **Talabalar** jadvalidan o‘chiradi.

2. DELETE FROM so'rovining tushuntirilishi:

DELETE FROM talabalar: Bu so‘rov **talabalar** jadvalidan ma'lumotlarni o‘chirishni anglatadi.

WHERE id = 1: Bu shart bo‘lib, faqat **id** ustuni **1** ga teng bo‘lgan talaba o‘chiriladi. Agar **WHERE** sharti bo‘lmasa, barcha talabalarning ma'lumotlari o‘chib ketadi, shuning uchun har doim **WHERE** shartini qo‘llash muhim.

3. Barcha ma'lumotlarni o‘chirish

Agar **talabalar** jadvalidagi barcha ma'lumotlarni o‘chirmoqchi bo‘lsangiz:

```
DELETE FROM talabalar;
```

75. MySQL MBBT da Universitet bazasi ichida talabalar jadvalidagi malumotlarni o’chirish uchun SQL so’rovini yozib bering. Bunda ID yangi kiritilganda yana 1 dan boshlab ketishini ta’minlang;

```
DELETE FROM talabalar;


ALTER TABLE talabalar AUTO_INCREMENT = 1;
```

DELETE FROM talabalar; — Bu so‘rov barcha yozuvlarni o‘chiradi.

ALTER TABLE talabalar AUTO_INCREMENT = 1; — Bu so‘rov **AUTO_INCREMENT** qiymatini 1 ga qaytaradi.

76. WHERE operatorining vazifasi nimadan iborat?

1. WHERE operatorining vazifasi:

- **Filtrlash: WHERE** operatori, ma'lumotlarni tanlab olishda shartli filtrlashni amalga oshiradi. U **SQL SELECT**, **UPDATE**, **DELETE** va boshqa so'rovlar bilan ishlatiladi.

- **Shartlar qo‘llash:** U =, **>, <, BETWEEN, IN**, **LIKE** va boshqa shart operatorlari bilan birgalikda ishlatiladi, shunday qilib kerakli yozuvlar bazadan tanlab olinadi.

2. Misollar:

Misol 1: SELECT so'rovida WHERE operatori bilan filtr

Agar **talabalar** jadvalidan **fakultet** ustuni "Informatika" bo'lgan talabalarning ma'lumotlarini olish kerak bo'lsa:

SELECT * FROM talabalar

WHERE fakultet = 'Informatika';

Bu so'rov **fakultet** ustuni "Informatika" bo‘lgan barcha talabalar ma'lumotlarini qaytaradi.

Misol 2: WHERE bilan AND operatori

Agar bir nechta shartni birlashtirish kerak bo'lsa, **AND** operatoridan foydalanish mumkin. Masalan, **kurs** 2 va **is_active** 1 bo‘lgan talabalarni olish:

SELECT * FROM talabalar

WHERE kurs = 2 AND is_active = 1;

Bu so‘rovda **kurs** 2 bo‘lgan va **is_active** 1 bo‘lgan talabalar tanlanadi.

77. Tadbirkorlik kompaniyasida orders va customers nomli ikkita jadval mavjud. Har bir buyurtma uchun mijozning ismi va buyurtma sanasini chiqarib beradigan so‘rov tuzing.

Mijozlarning ismi va buyurtma sanasini chiqarish uchun **INNER JOIN** operatori yordamida **orders** va **customers** jadvallarini **customer_id** ustuni orqali birlashtiramiz. So‘rov quyidagicha bo‘ladi:

SELECT customers.customer_name, orders.order_date

FROM orders

INNER JOIN customers ON orders.customer_id = customers.customer_id;

2. So'rovning tushuntirilishi:

- **SELECT customers.customer_name, orders.order_date**: Bu qismda biz **mijozning ismi** (customer_name) va **buyurtma sanasini** (order_date) chiqarishimiz kerak.
- **FROM orders**: **orders** jadvalidan boshlaymiz.
- **INNER JOIN customers ON orders.customer_id = customers.customer_id**: Bu joyda biz **orders** va **customers** jadvallarini **customer_id** ustuni orqali birlashtiramiz. **INNER JOIN** operatori faqat ikkala jadvalda ham mos yozuvlar mavjud bo‘lsa, natijalarni ko‘rsatadi.

3. Natija:

Bu so‘rov **orders** va **customers** jadvallaridan mijozlarning ismlari va buyurtma sanalarini chiqaradi. Har bir buyurtma uchun uning tegishli mijozining ismi va buyurtma sanasi ko‘rsatiladi.

Misol:

Agar **orders** jadvali quyidagi ma'lumotlarga ega bo'lsa:

order_id	customer_id	order_date
1	101	2025-05-01
2	102	2025-05-02
3	101	2025-05-03

Va **customers** jadvali quyidagicha bo'lsa:

customer_id	customer_name
101	Ali Rahmonov

78. Tadbirkorlik kompaniyasida orders va customers nomli ikkita jadval mavjud. Barcha mijozlar ro‘yxatini va agar ular buyurtma bergan bo‘lsa, tegishli buyurtma sanasini ham chiqarib beradigan so‘rov tuzing. (Buyurtma qilmagan mijozlar ham ro‘yxatda bo‘lishi kerak.)

SQL SO‘ROV:

SELECT customers.customer_name, orders.order_date

FROM customers

LEFT JOIN orders ON customers.customer_id = orders.customer_id;

TUSHUNTIRISH:

- **LEFT JOIN**: Bu JOIN turi **chap jadvaldagi barcha yozuvlarni (customers)** chiqaradi, hatto **o‘ng jadvalda (orders)** unga mos keladigan yozuv bo‘lmasa ham.
- **orders.order_date**: Agar mos yozuv mavjud bo‘lsa, bu ustunda buyurtma sanasi ko‘rsatiladi. Aks holda, bu qiymat **NULL** bo‘ladi.
- Shunday qilib, **har bir mijozning ismi** chiqadi, va agar mavjud bo‘lsa, uning **buyurtma sanasi** ham qo‘shiladi.

MISOL:

Agar **customers** jadvalda quyidagi yozuvlar bo‘lsa:

customer_id customer_name

1 Ali Rahmonov

2 Nodira Xamidova

3 Jamshid Karimov

Va **orders** jadvalda quyidagilar bo‘lsa:

order_id customer_id order_date

1 1 2025-05-01

2 2 2025-05-03

So‘rov natijasi quyidagicha bo‘ladi:

customer_name order_date

Ali Rahmonov 2025-05-01

79. Sizda products nomli jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak: Narxi 100 dan 500 gacha bo‘lgan (shu qiymatlar ham kiradi) mahsulotlarni tanlab oling.

SQL SO‘ROVI:

sql

КопироватьРедактировать

SELECT *

FROM products

WHERE price BETWEEN 100 AND 500;

TUSHUNTIRISH:

- **BETWEEN 100 AND 500** — bu price ustunidagi qiymatlar **100 ≥ price ≤ 500** oraliqda bo‘lishi kerakligini bildiradi.
- **BETWEEN** operatori **chegaralarni ham o‘z ichiga oladi** (ya'ni 100 va 500 qiymatlari ham tanlanadi).
- **SELECT *** — jadvaldagi barcha ustunlar chiqariladi. Agar faqat muayyan ustunlar kerak bo‘lsa, ularni nomma-nom yozish mumkin (masalan: product_name, price).

80. Sizda orders nomli jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak. Har bir mijoz tomonidan berilgan buyurtmalar sonini hisoblab, mijoz ID’si bo‘yicha guruhlang.

SQL SO‘ROVI:

SELECT customer_id, COUNT(order_id) AS total_orders

FROM orders

GROUP BY customer_id;

TUSHUNTIRISH:

- **COUNT(order_id)** – bu funksiyadan foydalanib, har bir mijoz nechta buyurtma berganini sanaymiz.
- **AS total_orders** – natijaga chiqadigan ustun nomini total_orders deb nomlaymiz.
- **GROUP BY customer_id** – barcha yozuvlarni customer_id bo‘yicha guruhlaymiz, ya’ni har bir mijoz uchun alohida natija bo‘ladi.

81. Sizda orders nomli jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak: Buyurtma miqdoriga qarab kategoriya belgilash: Agar amount 500 dan katta bo‘lsa, "Yuqori", Agar amount 100 dan 500 gacha bo‘lsa, "O‘rtacha", Agar amount 100 dan kichik bo‘lsa, "Past" deb belgilab, har bir buyurtma uchun ushbu kategoriyani ko‘rsatuvchi so‘rov yozing.

SQL SO‘ROVI (CASE operatori bilan):

SELECT order_id, amount,

```
  CASE
    WHEN amount > 500 THEN 'Yuqori'
    WHEN amount BETWEEN 100 AND 500 THEN 'O‘rtacha'
    WHEN amount < 100 THEN 'Past'
  END AS kategoriya
FROM orders;
```

TUSHUNTIRISH:

- **CASE** — bu SQLda shartli ifodalarni amalga oshiruvchi operator.
- **WHEN ... THEN** — har bir shartga mos ravishda natija belgilaydi.
- **AS kategoriya** — natijada chiqariladigan yangi ustun nomi.

MISOL:

Agar orders jadvalda quyidagicha yozuvlar bo‘lsa:

order_id	amount
1	700
2	300
3	50

So‘rov natijasi quyidagicha bo‘ladi:

order_id	amount	kategoriya
1	700	Yuqori
2	300	O‘rtacha

82. Sizda students va courses nomli ikkita jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak: Telefon raqami mavjud bo‘lgan va kamida bitta kursga yozilgan talabalarni tanlab oling.

1Agar courses jadvali har bir yozuvda student_id ni o‘z ichiga olsa:

```
SELECT DISTINCT s.student_id, s.student_name, s.phone
FROM students s
JOIN courses c ON s.student_id = c.student_id
WHERE s.phone IS NOT NULL AND s.phone <> '';
```

2.Agar alohida enrollments jadvali mavjud bo‘lsa (ko‘pdan-ko‘p aloqa uchun):

SELECT DISTINCT s.student_id, s.student_name, s.phone

FROM students s

JOIN enrollments e ON s.student_id = e.student_id

WHERE s.phone IS NOT NULL AND s.phone <> '';

TUSHUNTIRISH:

- **JOIN** — talabalarni kurslarga bog‘laydi.
- **s.phone IS NOT NULL AND s.phone <> ''** — faqat telefon raqami **bor** bo‘lgan talabalarni tanlaydi.
- **DISTINCT** — bir talaba bir nechta kursga yozilgan bo‘lsa, dublikat chiqmasligi uchun ishlatiladi.

83. Sizda students va courses nomli ikkita jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak: Yangi talaba qo‘shing va uni birdaniga bir nechta kursga yozing.

1.TALABANI QO‘SHISH (students jadvaliga):

INSERT INTO students (student_name, phone)

VALUES ('Ali Valiyev', '+998901234567');

Bu so‘rovdan so‘ng avtomatik student_id yaratiladi (agar AUTO_INCREMENT bo‘lsa).

2.So‘nggi qo‘shilgan talabaning student_idsini olish (MySQL uchun):

SET @student_id = LAST_INSERT_ID();

3.ALABANI KURSLARGA YOZISH (enrollments yoki bog‘lovchi jadvalga):

INSERT INTO enrollments (student_id, course_id)

VALUES

(@student_id, 1),

(@student_id, 3),

(@student_id, 5);

Bu yerda talaba course_id 1, 3, va 5 ga yozilmoqda.

TO‘LIQ KO‘RINISH:

```
-- 1. Talabani qo‘shish
INSERT INTO students (student_name, phone)
VALUES ('Ali Valiyev', '+998901234567');

-- 2. Oxirgi kiritilgan student_id ni olish
SET @student_id = LAST_INSERT_ID();
-- 3. Uni kurslarga yozish
INSERT INTO enrollments (student_id, course_id)
VALUES
  (@student_id, 1),
  (@student_id, 2),
  (@student_id, 4);
```

E’TIBOR BERING:

- enrollments jadvali quyidagicha bo‘lishi kerak:

```
CREATE TABLE enrollments (
  student_id INT,
  course_id INT,
  PRIMARY KEY (student_id, course_id),
  FOREIGN KEY (student_id) REFERENCES students(student_id),
  FOREIGN KEY (course_id) REFERENCES courses(course_id));
```

84. Sizda students, courses va grades nomli uchta jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak:Har bir talabaning kurs va baholari bo‘yicha ko‘rinish (VIEW) yarating.

SQL SO‘ROV (VIEW yaratish):

```
CREATE VIEW student_course_grades AS
SELECT
  s.student_name,
  c.course_name,
  g.grade
```

FROM grades g

JOIN students s ON g.student_id = s.student_id

JOIN courses c ON g.course_id = c.course_id;

TUSHUNTIRISH:

- grades jadvalidan boshlayapmiz, chunki u ikkala boshqa jadval bilan bog‘liq (student_id, course_id orqali).
- JOIN orqali students va courses jadvalidan kerakli ma’lumotlarni qo‘shyapmiz.
- **VIEW** nomi: student_course_grades — bu nom orqali keyin oddiy jadvaldek foydalanish mumkin.

KO‘RINISHDAN FOYDALANISH:

SELECT * FROM student_course_grades;

85. Sizda students, courses va grades nomli uchta jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak:Har bir talabaning kurs va baholari bo‘yicha ko‘rinish (VIEW) yarating. Talabaning ismi, telefon raqami, kurs nomi va olgan bahosi chiqishi kerak. students jadvali talabalar haqida, courses jadvali talabalar qaysi kursga yozilganligini, grades jadvali esa ularning baholarini saqlaydi.

SQL SO‘ROV: VIEW yaratish

CREATE VIEW student_course_grade_view AS

SELECT

  s.student_name,

  s.phone,

c.course_name,

g.grade

FROM grades g

JOIN students s ON g.student_id = s.student_id

JOIN courses c ON g.course_id = c.course_id;

TUSHUNTIRISH:

- **grades** jadvali asosiy bo‘lib xizmat qiladi, chunki unda **student_id** va **course_id** mavjud.
- **JOIN** orqali students jadvalidan **ism va telefon raqami**, courses jadvalidan esa **kurs nomi** olinadi.
- Natijada, har bir talabaga tegishli kurs va o‘sha kurs bo‘yicha bahosi ko‘rsatiladi.

VIEW dan foydalanish:

SELECT * FROM student_course_grade_view;

Bu orqali istalgan paytda to‘liq ko‘rinishni chaqirish mumkin.

86. Sizda students nomli jadval mavjud. Quyidagi talab bo‘yicha SQL so‘rovini yozish kerak:Ismida "Ali" so‘zi qatnashgan yoki telefon raqami "+99890" bilan boshlanadigan talabalarni tanlang.

SQL SO‘ROV:

SELECT *

FROM students

WHERE student_name LIKE '%Ali%'

OR phone LIKE '+99890%';

TUSHUNTIRISH:

- LIKE '%Ali%' → ismning istalgan qismida "Ali" bo‘lsa (boshi, o‘rtasi yoki oxiri).

- LIKE '+99890%' → telefon raqami +99890 bilan boshlansa.
- OR operatori → har ikkala shartdan biri to‘g‘ri bo‘lsa ham yetarli.

MISOL:

Agar students jadvalida quyidagi ma’lumotlar bo‘lsa:

student_id	student_name	phone
1	Alisher Karimov	+998901112233
2	Hasan Aliyev	+998933334455
3	Diyor Raxmatov	+998901234567

U holda **1, 2, 3-yozuvlar** ham tanlab olinadi, chunki:

- Alisher va Aliyev ismida "Ali" bor;
- Diyorning telefoni +99890 bilan boshlanadi.

87. C# da MySQL bilan bog‘lanish uchun qanday kutubxona ishlatiladi va bog‘lanish satri qanday yoziladi?

1. MySQL bilan bog‘lanish uchun kutubxona:

C# da MySQL bilan ishlash uchun **MySql.Data** kutubxonasidan foydalaniladi. Bu kutubxona MySQL serveriga bog‘lanish, so‘rovlar yuborish, ma’lumotlarni olish va boshqa operatsiyalarni amalga oshirish uchun kerak.

2. Kutubxonani o‘rnatish:

Agar siz **Visual Studio**da ishlayotgan bo‘lsangiz, **NuGet Package Manager** orqali MySql.Data paketini o‘rnatishingiz kerak:

Install-Package MySql.Data

Yoki **.NET CLI** orqali quyidagicha o‘rnatishingiz mumkin:

dotnet add package MySql.Data

3. Bog‘lanish satri (Connection String):

MySQL bilan bog‘lanish uchun quyidagi formatdagi **connection string** ishlatiladi:

```
string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
```

- **Server=localhost** — MySQL serverining manzili (agar lokal server bo‘lsa, localhost yoki 127.0.0.1).
- **Port=3306** — MySQL serverining port raqami (standart port).
- **Database=mydatabase** — bog‘lanmoqchi bo‘lgan ma’lumotlar bazasining nomi.
- **Uid=myusername** — MySQL foydalanuvchi nomi.
- **Pwd=mypassword** — foydalanuvchi paroli.

4. C# da bog‘lanish va so‘rov yuborish:

MySQL bilan bog‘lanish va so‘rov yuborishning oddiy namunasi:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
   public static void Main(string[] args)
   {
      string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
      // Bog‘lanishni yaratish
      using (MySqlConnection conn = new MySqlConnection(connectionString))
      {
         try
         {
            // Bog‘lanish
            conn.Open();
            Console.WriteLine("MySQL serverga bog‘landik!");
```

```
            // SQL so‘rovini yaratish
            string query = "SELECT * FROM students";
            MySqlCommand cmd = new MySqlCommand(query, conn)
            // So‘rovni bajarish
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                Console.WriteLine(reader["student_name"].ToString());
            }
            reader.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
}
```

88. C# da MySQL dan barcha talabalar ro‘yxatini o‘qish uchun qanday SQL so‘rovi va C# kod yozilishi kerak?

SQL So‘rovi:

Barcha talabalar ro‘yxatini olish uchun quyidagi SQL so‘rovini ishlatish mumkin:

SELECT * FROM students;

Ushbu so‘rov barcha talabalarning ma’lumotlarini (student_id, student_name, phone, va boshqa ustunlar) **students** jadvalidan olish uchun ishlatiladi.

C# Kod:

C# da MySQL dan barcha talabalar ro‘yxatini o‘qish uchun quyidagi kodni ishlatishingiz mumkin:

using MySql.Data.MySqlClient;

using System;

```
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL so‘rovini yaratish
                string query = "SELECT * FROM students";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // So‘rovni bajarish va natijalarni olish
                MySqlDataReader reader = cmd.ExecuteReader();
                // Har bir talaba ma’lumotlarini konsolga chiqarish
                while (reader.Read())
                {
                    string studentName = reader["student_name"].ToString();
                    string phone = reader["phone"].ToString();
                    Console.WriteLine("Talaba ismi: " + studentName + ", Telefon: " +
phone);
                }
                reader.Close();
```

```
            }
            catch (Exception ex)
            {
                Console.WriteLine("Xato: " + ex.Message);
            }
        }
    }
}
```

89. C# da foydalanuvchi kiritgan ism va yosh ma’lumotlarini MySQL jadvaliga qo‘shish uchun qanday kod yoziladi?

SQL So‘rovi:

Foydalanuvchi kiritgan ism va yosh ma'lumotlarini **students** jadvaliga qo‘shish uchun quyidagi SQL so‘rovini ishlatish mumkin:

```
INSERT INTO students (student_name, age)
VALUES ('Foydalanuvchi Ismi', 20);
```

Bu so‘rovda:

- **student_name** — talabani ismi (foydalanuvchi tomonidan kiritilgan);
- **age** — talabani yoshi (foydalanuvchi tomonidan kiritilgan).

C# Kod:

Foydalanuvchidan ism va yoshni olish va MySQL jadvaliga qo‘shish uchun quyidagi C# kodini ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Foydalanuvchidan ism va yoshni olish
        Console.Write("Ismingizni kiriting: ");
        string studentName = Console.ReadLine();
        Console.Write("Yoshingizni kiriting: ");
        int age = int.Parse(Console.ReadLine());
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL so‘rovini yaratish
                string query = "INSERT INTO students (student_name, age) VALUES
(@studentName, @age)";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // Parametrlarni qo‘shish
                cmd.Parameters.AddWithValue("@studentName", studentName);
                cmd.Parameters.AddWithValue("@age", age);
```

```
            // So‘rovni bajarish
            cmd.ExecuteNonQuery();
            Console.WriteLine("Yangi talaba qo‘shildi!");
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
  }
}
```

90. C# orqali student_id = 5 bo‘lgan talabaning yoshini 25 ga o‘zgartirish uchun qanday SQL UPDATE so‘rovi yoziladi va C# da qanday kod ishlatiladi?

SQL UPDATE So‘rovi:

student_id = 5 bo‘lgan talabaning yoshini 25 ga o‘zgartirish uchun quyidagi SQL **UPDATE** so‘rovini ishlatish mumkin:

```
UPDATE students
SET age = 25
WHERE student_id = 5;
```

- **students** — jadval nomi.
- **age** — o‘zgartiriladigan ustun (yosh).
- **student_id = 5** — faqat student_id qiymati 5 bo‘lgan talaba uchun yoshni o‘zgartirish.

C# Kod:

C# orqali yuqoridagi so‘rovni amalga oshirish uchun quyidagi kodni yozish mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL UPDATE so‘rovini yaratish
                string query = "UPDATE students SET age = 25 WHERE student_id =
5";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // So‘rovni bajarish
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Talabaning yoshi muvaffaqiyatli o‘zgartirildi!");
                }
                else
                {
```

```
                Console.WriteLine("Hech qanday yozuv o‘zgartirilmagan.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
  }
}
```

91. C# orqali student_id = 10 bo‘lgan talabani MySQL dan o‘chirish uchun qanday SQL DELETE so‘rovi yoziladi va bu C# da qanday ishlatiladi?

SQL DELETE So‘rovi:

student_id = 10 bo‘lgan talaba ma'lumotlarini **students** jadvalidan o‘chirish uchun quyidagi **DELETE** so‘rovini ishlatish mumkin:

```
DELETE FROM students
WHERE student_id = 10;
```

- **DELETE FROM students** — **students** jadvalidan yozuvni o‘chirish.
- **WHERE student_id = 10** — faqat student_id qiymati 10 bo‘lgan talaba yozuvi o‘chiriladi.

C# Kod:

C# orqali yuqoridagi so‘rovni amalga oshirish uchun quyidagi kodni yozish mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
   public static void Main(string[] args)
```

```
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL DELETE so‘rovini yaratish
                string query = "DELETE FROM students WHERE student_id = 10";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // So‘rovni bajarish
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Talaba muvaffaqiyatli o‘chirildi!");
                }
                else
                {
                    Console.WriteLine("Talaba topilmadi yoki o‘chirilgan yozuv mavjud
emas.");
                }
            }
            catch (Exception ex)
            {
```

```
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
}
}
```

92. C# orqali berilgan talabaning o‘rtacha bahosini MySQL bazasidan olish uchun qanday SQL so‘rovi yoziladi va bu C# da qanday amalga oshiriladi?

SQL So‘rovi:

Berilgan talabaning o‘rtacha bahosini olish uchun **grades** jadvalidan o‘rtacha baho (AVG) ni hisoblash uchun quyidagi SQL so‘rovini ishlatish mumkin:

```
SELECT AVG(grade) AS average_grade
FROM grades
WHERE student_id = @studentId;
```

- **AVG(grade)** — talabaning barcha baholarining o‘rtacha qiymatini hisoblaydi.
- **WHERE student_id = @studentId** — faqat berilgan talabaning baholarini hisoblash uchun **student_id** ni filtrlash.

C# Kod:

C# da berilgan talabaning o‘rtacha bahosini olish uchun quyidagi kodni ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
```

```
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Foydalanuvchidan student_id ni olish
        Console.Write("Talaba ID sini kiriting: ");
        int studentId = int.Parse(Console.ReadLine());
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL so‘rovini yaratish
                string query = "SELECT AVG(grade) AS average_grade FROM grades
WHERE student_id = @studentId";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // Parametrlarni qo‘shish
                cmd.Parameters.AddWithValue("@studentId", studentId);
                // So‘rovni bajarish va natijalarni olish
                object result = cmd.ExecuteScalar();

                if (result != DBNull.Value)
                {
                    Console.WriteLine($"Talaba ID {studentId} ning o'rtacha bahosi:
{result}");
                }
                else
```

```
            {
                Console.WriteLine($"Talaba ID {studentId} uchun baholar mavjud emas.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);
        }}}}
```

93. C# orqali har bir talabaning qaysi kurslarga yozilganini olish uchun qanday SQL JOIN so‘rovi yoziladi va bu C# da qanday amalga oshiriladi?

SQL JOIN So‘rovi:

Har bir talabaning qaysi kurslarga yozilganini olish uchun students, courses va grades jadvalarini **JOIN** qilishimiz kerak. Quyidagi SQL so‘rovi yordamida har bir talaba uchun tegishli kurslarni olish mumkin:

```
SELECT s.student_id, s.name AS student_name, c.course_name
FROM students s
JOIN grades g ON s.student_id = g.student_id
JOIN courses c ON g.course_id = c.course_id;
```

- **students** jadvali — talabaning ma'lumotlari saqlanadi.
- **courses** jadvali — kurslarning ma'lumotlari saqlanadi.
- **grades** jadvali — talabalar qaysi kurslarga yozilganini va ularning baholarini saqlaydi.
- **JOIN** — jadvalarni birlashtirish uchun ishlatiladi. Bu so‘rovda:
    - **students va grades** jadvali student_id bo‘yicha bog‘lanadi.
    - **grades va courses** jadvali course_id bo‘yicha bog‘lanadi.

C# Kod:

C# da har bir talabaning qaysi kurslarga yozilganini olish uchun quyidagi kodni ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
```

```
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL JOIN so‘rovini yaratish
                string query = @"
                    SELECT s.student_id, s.name AS student_name, c.course_name
                    FROM students s
                    JOIN grades g ON s.student_id = g.student_id
                    JOIN courses c ON g.course_id = c.course_id";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // So‘rovni bajarish
                MySqlDataReader reader = cmd.ExecuteReader();
                // Natijalarni chiqarish
                while (reader.Read())
                {
```

```
                int studentId = reader.GetInt32("student_id");
                string studentName = reader.GetString("student_name");
                string courseName = reader.GetString("course_name");
                Console.WriteLine($"Talaba: {studentName} (ID: {studentId}) kursga yozilgan: {courseName}");
            }
            reader.Close();
        }catch (Exception ex) {
            Console.WriteLine("Xato: " + ex.Message);}}}}
```

94. C# orqali ismida "Ali" so‘zi mavjud bo‘lgan barcha talabalarni MySQL bazasidan olish uchun qanday SQL so‘rovi yoziladi va bu C# da qanday amalga oshiriladi?

SQL So‘rovi:

Ismida "Ali" so‘zi mavjud bo‘lgan barcha talabalarni olish uchun quyidagi **SQL SELECT** so‘rovini ishlatish mumkin:

```
SELECT * FROM students
WHERE name LIKE '%Ali%';
```

- **LIKE '%Ali%'** — bu so‘rov name ustunidagi barcha qiymatlarni tekshiradi va ismida "Ali" so‘zi mavjud bo‘lgan barcha yozuvlarni qaytaradi. % belgilari qaysi joyda bo‘lishidan qat'i nazar "Ali"ni izlaydi.

C# Kod:

C# da ismida "Ali" so‘zi mavjud bo‘lgan talabalarni olish uchun quyidagi kodni ishlatishingiz mumkin:

```
csharp
КопироватьРедактировать
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
   public static void Main(string[] args)
   {
```

```
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL so‘rovini yaratish
                string query = "SELECT * FROM students WHERE name LIKE '%Ali
%'";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // So‘rovni bajarish
                MySqlDataReader reader = cmd.ExecuteReader();
                // Natijalarni chiqarish
                while (reader.Read())
                {
                    int studentId = reader.GetInt32("student_id");
                    string studentName = reader.GetString("name");
                    Console.WriteLine($"Talaba ID: {studentId}, Ism: {studentName}");
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Xato: " + ex.Message);
```

```
        }
    }
  }
}
```

95. C# orqali bazadan berilgan ID bo‘yicha talabani o‘chirish uchun qanday SQL DELETE so‘rovi yoziladi va bu C# da qanday ishlatiladi?

SQL DELETE So‘rovi:

Bazadan berilgan ID bo‘yicha talabani o‘chirish uchun **DELETE** so‘rovi quyidagicha yoziladi:

```
DELETE FROM students
WHERE student_id = @studentId;
```

- **DELETE FROM students** — **students** jadvalidan ma'lumotlarni o‘chirish.
- **WHERE student_id = @studentId** — faqat berilgan **student_id** bo‘yicha talaba o‘chiriladi. **@studentId** parametrli so‘rov bo‘lib, bu orqali ma'lumot xavfsizligi ta'minlanadi.

C# Kod:

C# da berilgan ID bo‘yicha talabani o‘chirish uchun quyidagi kodni ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
   public static void Main(string[] args)
   {
      // MySQL serveriga bog‘lanish uchun connection string
```

```
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Foydalanuvchidan student_id ni olish
        Console.Write("O'chirilishi kerak bo'lgan talabaning ID sini kiriting: ");
        int studentId = int.Parse(Console.ReadLine());
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL DELETE so‘rovini yaratish
                string query = "DELETE FROM students WHERE student_id =
@studentId";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // Parametrni qo‘shish
                cmd.Parameters.AddWithValue("@studentId", studentId);
                // So‘rovni bajarish
                int rowsAffected = cmd.ExecuteNonQuery();

                if (rowsAffected > 0)
                {
                    Console.WriteLine($"Talaba ID {studentId} muvaffaqiyatli
o‘chirildi.");
                }
                else
                {
```

```
                Console.WriteLine($"Talaba ID {studentId} topilmadi.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
}}
```

96. C# orqali student_id = 7 bo‘lgan talabaning yoshini 23 ga o‘zgartirish uchun qanday SQL UPDATE so‘rovi yoziladi va bu C# da qanday ishlatiladi?

SQL UPDATE So‘rovi:

UPDATE so‘rovini yozish orqali student_id = 7 bo‘lgan talabaning yoshini 23 ga o‘zgartirish uchun quyidagi SQL so‘rovi ishlatiladi:

```
UPDATE students
SET age = 23
WHERE student_id = 7;
```

- **UPDATE students** — **students** jadvalidagi yozuvlarni yangilash.
- **SET age = 23** — **age** ustunini yangi qiymatga, ya'ni 23 ga o‘zgartirish.
- **WHERE student_id = 7** — faqat **student_id = 7** bo‘lgan talabani yangilash.

C# Kod:

C# da student_id = 7 bo‘lgan talabaning yoshini 23 ga o‘zgartirish uchun quyidagi kodni ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
```

```
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // student_id = 7 bo‘lgan talabaning yoshini o‘zgartirish
        int studentId = 7;
        int newAge = 23;
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL UPDATE so‘rovini yaratish
                string query = "UPDATE students SET age = @newAge WHERE
student_id = @studentId";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // Parametrlarni qo‘shish
                cmd.Parameters.AddWithValue("@newAge", newAge);
                cmd.Parameters.AddWithValue("@studentId", studentId);
                // So‘rovni bajarish
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine($"Talaba ID {studentId} ning yoshi {newAge} ga
o‘zgartirildi.");
                }
```

```
            else
            {
                Console.WriteLine($"Talaba ID {studentId} topilmadi.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Xato: " + ex.Message);}}}}
```

97. C# orqali foydalanuvchi kiritgan ism va yosh bo‘yicha yangi talabani MySQL bazasiga qo‘shish uchun qanday kod yozish kerak?

SQL INSERT So‘rovi:

Foydalanuvchidan ism va yoshni olib, yangi talabani qo‘shish uchun **INSERT** so‘rovi quyidagicha yoziladi:

```
INSERT INTO students (name, age)
VALUES (@name, @age);
```

- **INSERT INTO students (name, age)** — **students** jadvaliga **name** va **age** ustunlariga yangi ma'lumot qo‘shish.
- **VALUES (@name, @age)** — foydalanuvchidan kiritilgan **name** va **age** qiymatlarini bazaga qo‘shish.

C# Kod:

C# da foydalanuvchidan ism va yoshni olib, yangi talabani MySQL bazasiga qo‘shish uchun quyidagi kodni ishlatishingiz mumkin:

```
csharp
КопироватьРедактировать
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
```

```
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Foydalanuvchidan ism va yoshni olish
        Console.Write("Talabaning ismini kiriting: ");
        string name = Console.ReadLine();
        Console.Write("Talabaning yoshini kiriting: ");
        int age = int.Parse(Console.ReadLine());
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
            try
            {
                // Bog‘lanishni ochish
                conn.Open();
                Console.WriteLine("MySQL serverga bog‘landik!");
                // SQL INSERT so‘rovini yaratish
                string query = "INSERT INTO students (name, age) VALUES (@name,
@age)";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                // Parametrlarni qo‘shish
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@age", age);
                // So‘rovni bajarish
                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Talaba muvaffaqiyatli qo‘shildi.");
                }
```

```
        else
        {
            Console.WriteLine("Talaba qo‘shishda xatolik yuz berdi.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Xato: " + ex.Message);}}}}
```

98. C# da berilgan ID bo‘yicha talabaning ismi va yoshi haqida ma’lumot olish uchun qanday SQL so‘rovi yoziladi va bu C# da qanday ishlatiladi?

SQL SELECT So‘rovi:

Berilgan **student_id** bo‘yicha talabaning **ismi** va **yoshi** haqida ma'lumot olish uchun **SELECT** so‘rovi quyidagicha yoziladi:

```
SELECT name, age
FROM students
WHERE student_id = @studentId;
```

- **SELECT name, age** — talabaning **name** va **age** ustunlari tanlanadi.
- **FROM students** — **students** jadvalidan ma'lumot olish.
- **WHERE student_id = @studentId** — **student_id** bo‘yicha filtrlanadi, bu yerda **@studentId** parametrli so‘rov bo‘lib, kiritilgan ID bo‘yicha ma'lumot olinadi.

C# Kod:

C# da berilgan **student_id** bo‘yicha talabaning ismi va yoshini olish uchun quyidagi kodni ishlatishingiz mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
   public static void Main(string[] args)
```

```
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        // Foydalanuvchidan student_id ni olish
        Console.Write("Talabaning ID sini kiriting: ");
       int studentId = int.Parse(Console.ReadLine());
        // Bog‘lanishni yaratish
        using (MySqlConnection conn = new MySqlConnection(connectionString))
        {
           try
           {
              conn.Open();
              Console.WriteLine("MySQL serverga bog‘landik!");
              string query = "SELECT name, age FROM students WHERE student_id
= @studentId";
              MySqlCommand cmd = new MySqlCommand(query, conn);
              cmd.Parameters.AddWithValue("@studentId", studentId);
              MySqlDataReader reader = cmd.ExecuteReader();
              if (reader.HasRows)
              {                   while (reader.Read())
                 {
                    string name = reader.GetString("name");
                    int age = reader.GetInt32("age");
                    Console.WriteLine($"Talaba: {name}, Yoshi: {age}");
                 }
              }
              else
              {
```

```
            Console.WriteLine($"ID {studentId} bo‘yicha talaba topilmadi.");
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Xato: " + ex.Message);
    }}}}
```

99. C# da MySQL ulanish xatolarini (try-catch) qanday aniqlash va foydalanuvchiga to‘g‘ri xabar berish mumkin?

MySQL ulanish xatolarini aniqlash va foydalanuvchiga xabar berish uchun C# kod:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        try
        {
            // Bog‘lanishni yaratish
            using (MySqlConnection conn = new
MySqlConnection(connectionString))
            {
                // Bog‘lanishni ochish
                conn.Open();
```

```
            Console.WriteLine("MySQL serverga bog‘landik!");
            // Boshqa amallarni bajarish
            // Masalan, SELECT so‘rovlarini bajarish va h.k.
        }
    }
    catch (MySqlException mysqlEx)
    {
        // MySQLga oid xatolarni ushlash
        Console.WriteLine($"MySQL xatolik: {mysqlEx.Message}");
        // Xatolik turi bo‘yicha to‘g‘ri xabar berish
        if (mysqlEx.Number == 1042)  // Serverga ulanish xatosi (hostname yoki IP noto‘g‘ri)
        {
            Console.WriteLine("MySQL serverga ulanishda xatolik. Server manzilingizni tekshiring.");
        }
        else if (mysqlEx.Number == 1045)  // Foydalanuvchi nomi yoki parol xatosi
        {
            Console.WriteLine("Foydalanuvchi nomi yoki parol noto‘g‘ri.");
        }
        else
        {
            // Umumiy MySQL xatolik xabari
            Console.WriteLine("Boshqa MySQL xatolik yuz berdi.");
        }
    }
    catch (Exception ex)
    {
        // Boshqa xatolarni ushlash (masalan, tarmoq xatoliklari)
```

```
            Console.WriteLine($"Umumiy xatolik: {ex.Message}");
        }
    }
}
```

100. C# orqali oxirgi qo‘shilgan talabaning ID sini olish uchun qanday SQL so‘rovi ishlatiladi?

SQL So‘rovi:

Oxirgi qo‘shilgan talabaning **ID** sini olish uchun SQL so‘rovi quyidagicha bo‘ladi:

```
SELECT LAST_INSERT_ID();
```

Bu so‘rov oxirgi qo‘shilgan yozuvning **auto-increment** maydonini (masalan, student_id) qaytaradi.

C# Kod:

C# da oxirgi qo‘shilgan talabaning **ID** sini olish uchun quyidagi kodni ishlatish mumkin:

```
using MySql.Data.MySqlClient;
using System;
public class MySQLExample
{
    public static void Main(string[] args)
    {
        // MySQL serveriga bog‘lanish uchun connection string
        string connectionString =
"Server=localhost;Port=3306;Database=mydatabase;Uid=myusername;Pwd=mypa
ssword;";
        try
        {
            // Bog‘lanishni yaratish
```

```
        using (MySqlConnection conn = new
MySqlConnection(connectionString))
        {
            // Bog‘lanishni ochish
            conn.Open();
            Console.WriteLine("MySQL serverga bog‘landik!");
            // Yangi talabani qo‘shish (INSERT)
            string insertQuery = "INSERT INTO students (name, age) VALUES
('Ali', 22)";
            MySqlCommand cmdInsert = new MySqlCommand(insertQuery, conn);
            cmdInsert.ExecuteNonQuery();  // INSERT so‘rovini bajarish
            // Oxirgi qo‘shilgan talabaning ID sini olish
            string selectQuery = "SELECT LAST_INSERT_ID()";
            MySqlCommand cmdSelect = new MySqlCommand(selectQuery, conn);
            object result = cmdSelect.ExecuteScalar();  // Oxirgi ID ni olish
            if (result != null)
            {
                int lastInsertedId = Convert.ToInt32(result);
                Console.WriteLine($"Oxirgi qo‘shilgan talabaning ID:
{lastInsertedId}");
            }
            else
            {
                Console.WriteLine("ID olishda xatolik yuz berdi.");
            }
        }
    }
    catch (Exception ex)
    {
        // Xatolikni ushlash
```

```
            Console.WriteLine("Xato: " + ex.Message);
        }
    }
}
```