1. **Что такое Prisma?**

**Prisma - это современный ORM (Object-Relational Mapping) и инструмент для работы с базами данных в среде Node.js, TypeScript и других экосистем. Он предоставляет удобный способ создания типобезопасных запросов к базам данных с помощью простого и выразительного API. Prisma обеспечивает автоматическую генерацию кода на основе схемы базы данных, что позволяет разработчикам быстро и легко взаимодействовать с данными.**

1. **Назовите отличия Prisma от других фреймворков ORM?**

* Автоматическую генерацию типобезопасного клиентского API на основе схемы базы данных.
* Нативную поддержку для множества баз данных, включая PostgreSQL, MySQL, SQLite и Microsoft SQL Server.
* Простой и выразительный API для создания и выполнения запросов к базе данных.
* Встроенную поддержку для сложных операций, таких как транзакции и агрегированные функции.
* Возможность работы с базами данных как на локальном компьютере, так и в облаке.

1. **Какова роль Prisma Client в доступе к данным и управлении ими?**

**Prisma Client - это сгенерированный автоматически клиентский код, который обеспечивает типобезопасный доступ к данным в базе данных. Он предоставляет методы для выполнения различных операций с данными, таких как создание, чтение, обновление и удаление записей. Prisma Client также обеспечивает возможность работы с транзакциями и выполнения сложных запросов.**

1. **Можно ли использовать Prisma с базами данных SQL и NoSQL?**

**Prisma** может использоваться с базами данных SQL и NoSQL. [Он поддерживает такие базы данных, как PostgreSQL, MySQL, SQL Server, SQLite, MongoDB, CockroachDB и другие](https://www.prisma.io/docs/orm/reference/supported-databases)

1. **Какие языки программирования совместимы с Prisma?**

Prisma предназначен для использования с JavaScript и TypeScript, особенно в среде Node.js.

1. **Расскажите, где и как можно использовать *every*, *none* и *some*?**

В **Prisma**, every, none и some[используются для фильтрации данных на основе отношенийevery возвращает записи, где все связанные записи удовлетворяют условию, none возвращает записи, где ни одна из связанных записей не удовлетворяет условию, а some возвращает записи, где хотя бы одна из связанных записей удовлетворяет условию](https://www.prisma.io/docs/orm/prisma-client/queries/relation-queries" \t "_blank)

1. **Для чего нужен метод *upsert*?**

[Метод **upsert** в **Prisma** используется для обновления записи, если она существует, или создания новой записи, если она не существует](https://stackoverflow.com/questions/73905852/how-to-use-upsert-with-prisma)

1. **Как включить регистронезависимый поиск?**

[Для включения регистронезависимого поиска в **Prisma**, вы можете использовать свойство mode: 'insensitive' на по-полевой основе при фильтрации](https://www.prisma.io/docs/orm/prisma-client/queries/case-sensitivity)

1. **Объясните назначение методов *$queryRaw*, *$executeRaw*.**

Методы **$queryRaw** и **$executeRaw** в **Prisma**[используются для выполнения сырых SQL-запросов$queryRaw возвращает фактические записи (например, используя SELECT), а $executeRaw возвращает количество затронутых строк (например, после UPDATE или DELETE)](https://www.prisma.io/docs/orm/prisma-client/queries/raw-database-access/raw-queries" \t "_blank)

1. **Какие виды транзакций существуют в Prisma?**

[**Prisma** поддерживает различные типы транзакций, включая вложенные записи, пакетные / массовые транзакции и интерактивные транзакции](https://www.prisma.io/docs/orm/prisma-client/queries/transactions)