SQL-Interview auf Englisch, Deutsch und Türkisch

Q1: What is the **difference** between **SQL** and **MySQL**?

A1: 1. SQL is a standard language which stands for Structured Query Language based on the English language. MySQL is a database management system.

2. SQL is the core of the relational database which is used for accessing and managing database but MySQL is an RDMS (Relational Database Management System) such as SQL Server, Informix etc.

Q2: What are the **different subsets** of SQL?

A2: 1. DDL (Data Definition Language)

2. DML (Data Manipulation Language)

3. DCL (Data Control Language)

Q1: Was ist der Unterschied zwischen **SQL** und **MySQL**?

A1: 1. SQL ist eine Standardsprache, die für Structured Query Language steht und auf der englischen Sprache basiert. MySQL ist ein Datenbankverwaltungssystem.;

2. SQL ist der Kern der relationalen Datenbank, die für den Zugriff auf und die Verwaltung von Datenbanken verwendet wird. MySQL ist jedoch ein RDMS (Relational Database Management System) wie SQL-Server, Informix usw.

F2: Was sind die verschiedenen Teilmengen von SQL?

A2: 1. DDL (Data Definition Language)

2. DML (Data Manipulation Language)

3. DCL (Data Control Language)

S1: **SQL** ile **MySQL** arasındaki fark nedir?

C1: 1. SQL, İngilizceye dayalı Yapılandırılmış Sorgu Dili anlamına gelen standart bir dildir. MySQL bir veritabanı yönetim sistemidir.

2. SQL, veritabanına erişmek ve yönetmek için kullanılan ilişkisel veritabanının çekirdeğidir ancak MySQL, SQL Server, Informix vb. gibi bir RDMS'dir (İlişkisel Veritabanı Yönetim Sistemi).

S2: SQL 'in farklı alt kümeleri nelerdir?

C2: 1. DDL (Data Definition Language- Veri Tanımlama Dili)

2. DML (Data Manipulation Language- Veri İşleme Dili)

3. DCL (Data Control Language- Veri Kontrol Dili)

Q3: What do you mean by **table** and **field** in SQL?

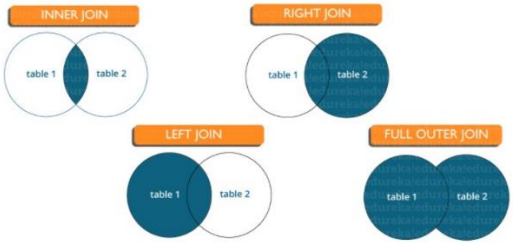
A3: A table refers to a collection of data in an organized manner in form of rows and columns. A field refers to the number of columns in a table.

For example: **Table**: Student_Information **Field**: Stu_Id, Stu_Name, Stu_Marks

Q4: What are **joins** in SQL?

A4: A JOIN clause is used to combine rows from two or more tables, based on a related column between them. It is used to merge two tables or retrieve data from there. There are 4 joins in SQL namely:

1. Inner Join
2. Right Join
3. Left Join
4. Full Join



F3: Was meinen Sie mit **Tabelle** und **Feld** in SQL?

A3: Eine Tabelle bezieht sich auf eine Sammlung von Daten in organisierter Form in Form von Zeilen und Spalten. Ein Feld bezieht sich auf die Anzahl der Spalten in einer Tabelle.

Beispiel: **Table**: Student_Information **Field**: Stu_Id, Stu_Name, Stu_Marks

F4: Was sind **Joins** in SQL?

A4: Eine JOIN-Klausel wird verwendet, um Zeilen aus zwei oder mehr Tabellen basierend auf einer zugehörigen Spalte zwischen ihnen zu kombinieren. Es wird verwendet, um zwei Tabellen zusammenzuführen oder Daten von dort abzurufen. Es gibt 4 Joins in SQL, nämlich:

1. Inner Join
2. Right Join
3. Left Join
4. Full Join

S3: SQL'de **table** ve **field** derken neyi kastediyorsunuz?

C3: Bir tablo, satırlar ve sütunlar şeklinde organize bir şekilde bir veri koleksiyonunu ifade eder. Alan, bir tablodaki sütunların sayısını ifade eder.

Örnek: **Table**: Student_Information **Field**: Stu_Id, Stu_Name, Stu_Marks

S4: SQL de Joins ler nelerdir?

C4: Bir JOIN yan tümcesi, iki veya daha fazla tablodan satırları aralarındaki ilgili sütuna göre birleştirmek için kullanılır. İki tabloyu birleştirmek veya oradan veri almak için kullanılır. SQL'de 4 birleştirme vardır:

1. Inner Join
2. Right Join
3. Left Join
4. Full Join

Q5: What is the **difference** between CHAR and VARCHAR2 **datatype** in SQL?

A5:

Both Char and Varchar2 are used for characters datatype but varchar2 is used for character strings of variable length whereas Char is used for strings of fixed length.

For example, char(10) can only store 10 characters and will not be able to store a string of any other length whereas varchar2(10) can store any length i.e. 6,8,2 in this variable.

F5: Was ist der **Unterschied** zwischen dem **Datentyp** CHAR und VARCHAR2 in SQL?

A5:

Sowohl Char als auch Varchar2 werden für Zeichendatentypen verwendet, aber varchar2 wird für Zeichenfolgen variabler Länge verwendet, während Char für Zeichenfolgen fester Länge verwendet wird.

Beispielsweise kann char(10) nur 10 Zeichen speichern und kann keine Zeichenfolge mit einer anderen Länge speichern, während varchar2 (10) eine beliebige Länge speichern kann, d. H. 6,8,2 in dieser Variablen.

S5: SQL'de CHAR ve VARCHAR2 **veri türü** (**datatype**) arasındaki **fark** nedir?

C5: Karakter veri türü için hem char hem de Varchar2 kullanılır, ancak değişken uzunluktaki karakter dizeleri için varchar2 kullanılırken, sabit uzunluktaki dizeler için char kullanılır.

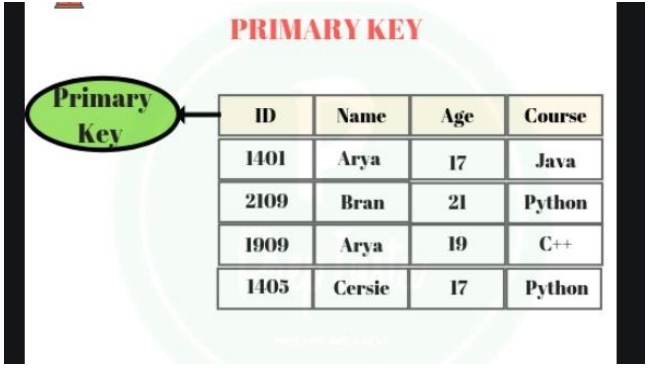
Örneğin, char(10) yalnızca 10 karakter saklayabilir ve başka uzunlukta bir dizeyi saklayamaz, oysa varchar2 (10) bu değişkende 6,8,2 gibi herhangi bir uzunluğu depolayabilir.

Q6: What is a **Primary key**?

A6: A Primary key is a column (or collection of columns) or a set of columns that uniquely identifies each row in the table.

Uniquely identifies a single row in the table.

Null values not allowed.



ID	Name	Age	Course
1401	Arya	17	Java
2109	Bran	21	Python
1909	Arya	19	C++
1403	Cersie	17	Python

F6: Was ist ein **Primary key**?

A6: Ein Primärschlüssel ist eine Spalte (oder eine Sammlung von Spalten) oder eine Reihe von Spalten, die jede Zeile in der Tabelle eindeutig identifiziert.

Identifiziert eine einzelne Zeile in der Tabelle eindeutig.

Nullwerte nicht erlaubt.

S6: **Primary key** nedir?

C6: Primary key tablodaki her satırı benzersiz bir şekilde tanımlayan bir sütun (veya sütun koleksiyonu) veya bir dizi sütundur.

Tablodaki tek bir satırı benzersiz şekilde tanımlar.

Boş değerlere izin verilmez.

Q7: What are **Constraints**?

A7: Constraints are used to specify the limit on the data type of the table.

NOT NULL

CHECK

DEFAULT

UNIQUE

PRIMARY KEY

FOREIGN KEY

F7: Was sind Einschränkungen (**Constraints**)?

A7: Einschränkungen (**Constraints**) werden verwendet, um die Grenze für den Datentyp der Tabelle anzugeben.

NOT NULL

CHECK

DEFAULT

UNIQUE

PRIMARY KEY

FOREIGN KEY

S7: Constraints (kısıtlamalar) nelerdir?

C7: Constraints tablonun veri türü üzerindeki sınırı belirtmek için kullanılır.

NOT NULL

CHECK

DEFAULT

UNIQUE

PRIMARY KEY

FOREIGN KEY

Q8: What is the difference between **DELETE** and **TRUNCATE** statements?

A8:

Delete command is used to delete a row in a table. Truncate is used to delete all the rows from a table.

You can rollback data after using delete statement. You cannot rollback data after using Truncate.

Delete is a DML command. Truncate is a DDL command.

Delete is slower than truncate statement. Truncate is faster.

F8: Was ist der Unterschied zwischen **DELETE**- und **TRUNCATE**-Anweisungen (Statements)?

A8: DELETE wird verwendet, um eine Zeile in einer Tabelle zu löschen. TRUNCATE wird verwendet, um alle Zeilen aus einer Tabelle zu löschen.

Sie können Daten nach Verwendung von DELETE zurücksetzen. Sie können nach Verwendung von TRUNCATE keine Daten zurücksetzen.

Delete ist ein DML-Command. Truncate ist ein DDL-Command.

Delete ist langsamer als Truncate-Statements. Truncate ist schneller.

S8: **DELETE** (silme) ve **TRUNCATE** (kesme) arasındaki farklar nelerdir?

C8: Sil komutu, tablodaki bir satırı silmek için kullanılır. Truncate, bir tablodaki tüm satırları silmek için kullanılır.

Delete ifadesini kullandıktan sonra verileri geri alabilirsiniz. Truncate kullandıktan sonra verileri geri alamazsınız.

Delete bir DML komutudur. Truncate bir DDL komutudur.

Delete, Truncate ifadesinden daha yavaştır. Truncate daha hızlıdır.

Q9: What is a **Unique key**?

A9: Uniquely identifies a single row in the table.

Multiple values allowed per table.

Null values allowed.

Q10: What is a **Foreign key**?

A10: Foreign key maintains referential integrity by enforcing a link between the data in two tables.

The foreign key in the child table references the primary key in the parent table.

The foreign key constraint prevents actions that would destroy links between the child and parent tables.

F9: Was ist ein **Unique** (**Key**) Schlüssel?

A9: Identifiziert eine einzelne Zeile in der Tabelle eindeutig.

Pro Tabelle sind mehrere Werte zulässig.

Nullwerte erlaubt.

F10: Was ist ein **Foreign key** (**Fremdschlüssel**)?

A10: Der Fremdschlüssel behält die referenzielle Integrität bei, indem eine Verknüpfung zwischen den Daten in zwei Tabellen erzwungen wird.

Der Fremdschlüssel in der untergeordneten Tabelle verweist auf den Primärschlüssel in der übergeordneten Tabelle.

Die Fremdschlüsseleinschränkung verhindert Aktionen, die Verknüpfungen zwischen der untergeordneten und der übergeordneten Tabelle zerstören würden.

S9: **Unique key** nedir?

C9: Tablodaki tek bir satırı benzersiz şekilde tanımlar.

Tablo başına birden çok değere izin verilir.

Boş değerlere izin verilir.

S10: **Foreign Key** nedir?

C10: Foreign Key, iki tablodaki veriler arasında bir bağlantı oluşturarak bilgi tutarlılığını korur.

Alt tablodaki Foreign key, üst tablodaki Primary key başvurur. Foreign key kısıtlaması, alt ve üst tablolar arasındaki bağlantıları yok edecek eylemleri engeller.

Q11: What do you mean by **data integrity**?

A11: Data Integrity defines the accuracy as well as the consistency of the data stored in a database. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

F11: Was meinen Sie mit Datenintegrität (**data integrity**)?

A11: Datenintegrität definiert die Genauigkeit sowie die Konsistenz der in einer Datenbank gespeicherten Daten. Außerdem werden Integritätsbeschränkungen definiert, um Geschäftsregeln für die Daten durchzusetzen, wenn diese in eine Anwendung oder Datenbank eingegeben werden.

S11: Veri bütünlüğü (**data integrity**) ile neyi kastediyorsunuz?

C11: Veri Bütünlüğü, bir veritabanında depolanan verilerin tutarlılığını ve doğruluğunu tanımlar. Ayrıca, bir uygulamaya veya veritabanına girildiğinde veriler üzerinde iş kuralları uygulamak için bütünlük kısıtlamalarını da tanımlar.

Q12: What is the difference between **clustered** and **non-clustered index** in SQL?

A12: Clustered index is used for easy retrieval of data from the database and its faster whereas reading from non-clustered index is relatively slower.

F12: Was ist der Unterschied zwischen **clustered** and **non-clustered Index** in SQL?

A12: Der clustered Index wird zum einfachen Abrufen von Daten aus der Datenbank verwendet und ist schneller, während das Lesen aus dem non-clustered Index relativ langsamer ist.

S12: SQL'de kümelenmiş (**clustered**) ve kümelenmemiş (**non-clustered**) dizin (**index**) arasındaki fark nedir?

C12: Kümelenmiş dizin, veri tabanından verilerin kolayca alınması için kullanılır ve daha hızlıdır, öte yandan kümelenmemiş dizinden okumak nispeten yavaştır.

Q13: How write a SQL **query** to display the current **date**?

A13: In SQL, there is a built-in function called **GetDate()** which helps to return the current timestamp/date.

F13: Wie schreibe ich eine SQL-Query, um das aktuelle Datum anzuzeigen?

A13: In SQL gibt es eine integrierte Funktion namens GetDate(), mit deren Hilfe der aktuelle Zeitstempel / das aktuelle Datum zurückgegeben werden kann.

S13: Geçerli **tarihi görüntülemek** için bir SQL sorgusu nasıl yazılır?

C13: SQL'de, geçerli zaman damgasını/tarihi döndürmeye yardımcı olan **GetDate()** adında yerleşik bir işlev vardır.

Q14: What is The **Left Join**?

A14: Left Join in MySQL is used to return all the rows from the left table but only the matching rows from the right table where the join condition is fulfilled.

F14: Was ist der **Left Join**?

A14: Left Join in MySQL wird verwendet, um alle Zeilen aus der linken Tabelle zurückzugeben, aber nur die übereinstimmenden Zeilen aus der rechten Tabelle, in denen die Join-Condition erfüllt ist.

S14: Left Join nedir?

C14: MySQL ‘de Left Join, soldaki tablodan tüm satırları döndürmek için kullanılır, ancak yalnızca birleştirme koşulunun yerine getirildiği sağ tablodaki eşleşen satırları döndürür.

Q15: What do you mean by **Denormalization**?

A15: Denormalization refers to a technique which is used to access data from higher to lower forms of a database. It helps the database managers to increase the performance of the entire infrastructure as it introduces redundancy into a table.

F15: Was meinst du mit Denormalisierung?

A15: Denormalisierung bezieht sich auf eine Technik, mit der auf Daten von höheren zu niedrigeren Formen einer Datenbank zugegriffen wird. Es hilft den Datenbankmanagern, die Leistung der gesamten Infrastruktur zu steigern, da Redundanz in eine Tabelle eingeführt wird.

S15: Denormalizasyon ile neyi kastediyorsunuz?

C15: Denormalizasyon, bir veri tabanının yüksek ve düşük formlarından verilere erişmek için kullanılan bir tekniği ifade eder. Bir tabloya fazlalık getirerek veritabanı yöneticilerinin tüm altyapının performansını artırmasına yardımcı olur.

Q16: What are **Entities** and **Relationships**?

A16: Entities: A person, place, or thing in the real world about which data can be stored in a database. Tables store data that represents one type of entity. Relationships: Relation or links between entities that have something to do with each other.

F16: Was sind **Entities** (Entitäten) and **Relationships** (Beziehungen)?

A16: Entities: Eine Person, ein Ort oder eine Sache in der realen Welt, über die Daten in einer Datenbank gespeichert werden können. Tabellen speichern Daten, die einen Entitätstyp darstellen. Relationships: Beziehung oder Verknüpfungen zwischen Entitäten, die etwas miteinander zu tun haben.

S16: Varlıklar (**Entities**) ve İlişkiler (**Relationships**) nedir?

C16: Varlıklar: Verilerin bir veritabanında depolanabileceği hakkında gerçek dünyadaki bir kişi, yer veya şey. Tablolar, bir tür varlığı temsil eden verileri depolar. İlişkiler: Birbirleriyle bir ilgisi olan varlıklar arasındaki ilişki veya bağlantılardır.

Q17: Give an example for **Entities**?

A17: For example – A bank database has a customer table to store customer information. Customer table stores this information as a set of attributes (columns within the table) for each customer.

F17: Geben Sie ein Beispiel für **Entities**?

A17: Beispiel: Eine Bankdatenbank verfügt über eine Kundentabelle zum Speichern von Kundeninformationen. In der Kundentabelle werden diese Informationen als eine Reihe von Attributen (Spalten in der Tabelle) für jeden Kunden gespeichert.

S17: **Entities** için bir örnek veriniz?

C17: Örneğin; Bir banka veritabanında, müşteri bilgilerini depolamak için bir müşteri tablosu bulunur. Müşteri tablosu, bu bilgileri her müşteri için bir dizi öznitelik (tablodaki sütunlar) olarak depolar.

Q18: Give an example for **Relations**?

A18: For example – The customer name is related to the customer account number and contact information, which might be in the same table. There can also be relationships between separate tables (for example, customer to accounts).

F18: Geben Sie ein Beispiel für **Relations** (Beziehungen)?

A18: Beispiel: Der Kundenname bezieht sich auf die Kundenkontonummer und die Kontaktinformationen, die sich möglicherweise in derselben Tabelle befinden. Es kann auch Beziehungen zwischen separaten Tabellen geben (z. B. Kunde zu Konten).

S18: **Relations** için bir örnek veriniz?

C18: Örneğin- Müşteri adı, aynı tabloda bulunabilecek müşteri hesap numarası ve iletişim bilgileriyle ilgilidir. Ayrı tablolar arasında da ilişkiler olabilir (örneğin, müşteriden hesaplara).

F19: Was ist ein **Index**?

A19: An index refers to a performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and hence it will be faster to retrieve data.

Q20: Explain different **types of index**.

A20: There are three types of index namely:

Unique Index,

Clustered Index,

Non-Clustered Index.

F19: War es ein **Index**?

A19: Ein Index bezieht sich auf eine Methode zur Leistungsoptimierung, mit der Datensätze schneller aus der Tabelle abgerufen werden können. Ein Index erstellt einen Eintrag für jeden Wert und daher ist das Abrufen von Daten schneller.

F20: Erklären Sie **verschiedene Arten von Index**.

A20: Es gibt drei Arten von Index:

Unique Index,

Clustered Index,

Non-Clustered Index.

S19: **Index** nedir?

C19: Bir indeks, tablodan kayıtların daha hızlı alınmasına izin veren bir performans ayarlama yöntemini ifade eder. Bir dizin, her değer için bir giriş oluşturur ve bu nedenle verileri almak daha hızlı olacaktır.

S20: **Index çeşitlerini** açıklayınız?

C20: Üç tür index dizin vardır:

Unique Index, (Benzersiz Dizin)

Clustered Index, (Kümelenmiş Dizin)

Non-Clustered Index. (Kümelenmemiş Dizin)

Q21: What is **Normalization** and what are the **advantages** of it?

A21: Normalization is the process of organizing data to avoid **duplication** and **redundancy**. Some of the advantages are:

- Better Database organization
- More Tables with smaller rows
- Efficient data access
- Greater Flexibility for Queries
- Quickly find the information
- Easier to implement Security
- Allows easy modification

F21: Was ist **Normalisierung** (**Normalization**) und welche **Vorteile** bietet sie?

A21: Bei der Normalisierung werden Daten organisiert, um **Doppelarbeit** und **Redundanz** zu vermeiden. Einige der Vorteile sind:

- Bessere Datenbankorganisation
- Mehr Tabellen mit kleineren Zeilen
- Effizienter Datenzugriff
- Höhere Flexibilität für Abfragen
- Finden Sie schnell die Informationen
- Einfachere Implementierung von Sicherheit
- Ermöglicht eine einfache Änderung

S21: **Normalization** (Normalleştirme) nedir ve **avantajları** nelerdir?

C21: Normalleştirme, yinelemeyi (Duplication) ve fazlalığı (redundancy) önlemek için verileri düzenleme işlemidir. Avantajlardan bazıları şunlardır:

- Daha iyi Veritabanı organizasyonu
- Daha küçük satırlara sahip daha fazla Tablo
- Verimli veri erişimi
- Sorgular için Daha Fazla Esneklik
- Bilgileri hızlıca bulun
- Güvenliği uygulamak daha kolay
- Kolay değiştirmeye izin verir

Q22: What is the difference between **DROP** and **TRUNCATE** commands?

A22: DROP command removes a table and it cannot be rolled back from the database whereas TRUNCATE command removes all the rows from the table.

F22: Was ist der Unterschied zwischen DROP- und TRUNCATE-Command?

A22: Der Command DROP entfernt eine Tabelle und kann nicht aus der Datenbank zurückgesetzt werden, während der Command TRUNCATE alle Zeilen aus der Tabelle entfernt.

S22: DROP ve TRUNCATE komutları arasındaki fark nedir?

C22: DROP komutu bir tabloyu kaldırır ve veritabanından geri alınamaz, TRUNCATE komutu ise tablodan tüm satırları kaldırır.

Q23: Explain different **types of Normalization**.

A23:

First Normal Form (**1NF**) => No repeating groups within rows.

Second Normal Form (**2NF**) => Every non-key (supporting) column value is dependent on the whole primary key.

Third Normal Form (**3NF**) => Dependent solely on the primary key and no other non-key (supporting) column value.

F23: Erklären Sie **verschiedene Arten der Normalisierung**.

A23:

Erste Normalform (**1NF**) => Keine sich wiederholenden Gruppen innerhalb von Zeilen.

Zweite Normalform (**2NF**) => Jeder nicht unterstützende (unterstützende) Spaltenwert ist vom gesamten Primärschlüssel abhängig.

Dritte Normalform (**3NF**) => Abhängig ausschließlich vom Primärschlüssel und keinem anderen (unterstützenden) Spaltenwert ohne Schlüssel.

S23: **Farklı Normalleştirme türlerini** açıklayın.

C23:

İlk Normal Form (**1NF**) => Satırlarda yinelenen grup yok.

İkinci Normal Form (**2NF**) => Anahtar olmayan (destekleyen) her sütun değeri, tüm birincil anahtara bağlıdır.

Üçüncü Normal Form (**3NF**) => Yalnızca birincil anahtara bağlıdır ve anahtar olmayan (destekleyen) başka sütun değeri yoktur.

Q24: What is **ACID property** in a database?

A24: **ACID** stands for **Atomicity**, **Consistency**, **Isolation**, **Durability**. It is used to ensure that the data transactions are processed reliably in a database system.

Q25: Explain them?

A25:

Atomicity: It means if one part of any transaction fails, the entire transaction fails and the database state is left unchanged.

Consistency: you can say that your transaction never leaves the database without completing its state.

Isolation: The main goal of isolation is concurrency control.

Durability: it means that if a transaction has been committed, it will occur whatever may come in between such as power loss, crash or any sort of error.

F24: Was ist die **ACID-Eigenschaft** (**property**) in einer Datenbank (Database)?

A24: ACID steht für Atomizität (**Atomicity**), Konsistenz (**Consistency**), Isolation (**Isolation**), Haltbarkeit (**Durability**). Damit wird sichergestellt, dass die Datentransaktionen in einem Datenbanksystem zuverlässig verarbeitet werden.

F25: Erklären Sie sie?

A25:

Atomizität: Wenn ein Teil einer Transaktion fehlschlägt, schlägt die gesamte Transaktion fehl und der Datenbankstatus bleibt unverändert.

Konsistenz: Sie können sagen, dass Ihre Transaktion die Datenbank niemals verlässt, ohne ihren Status zu vervollständigen.

Isolation: Das Hauptziel der Isolation ist die Parallelitätskontrolle.

Haltbarkeit: Wenn eine Transaktion festgeschrieben wurde, tritt alles auf, was dazwischen liegt, z. B. Stromausfall, Absturz oder jede Art von Fehler.

S24: Bir veri tabanındaki (Database) **ACID** özelliği (**property**) nedir?

C24: ACID, Atomiklik (**Atomicity**), Tutarlılık (**Consistency**), İzolasyon (**Isolation**), Dayanıklılık (**Durability**) anlamına gelir. Veri işlemlerinin bir veritabanı sisteminde güvenilir bir şekilde işlenmesini sağlamak için kullanılır.

S25: Bunları açıklayınız?

C25:

Atomiklik: Herhangi bir işlemin bir kısmının başarısız olması, tüm işlemin başarısız olması ve veritabanı durumunun değişmeden bırakılması anlamına gelir.

Tutarlılık: İşleminizin durumunu tamamlamadan asla veritabanından çıkmadığını söyleyebilirsiniz.

İzolasyon: İzolasyonun temel amacı eşzamanlılık kontrolüdür.

Dayanıklılık: Bir işlem gerçekleştirilmişse, güç kaybı, çökme veya herhangi bir hata gibi arada olabilecek ne olursa olsun gerçekleşeceği anlamına gelir.

Q26: What do you mean by “**Trigger**” in SQL?

A26: Trigger in SQL, it allows you to execute a batch of code when an insert, update or any other query is executed against a specific table.

S26: SQL'de **Trigger** (Tetikleyici) ile neyi kastediyorsunuz?

C26: Trigger belirli bir tabloya karşı bir ekleme, güncelleme veya başka herhangi bir sorgu yürütüldüğünde, bir toplu kod çalıştırmanıza olanak tanır.

F26: Was meinst du mit "**Trigger**" in SQL?

A26: Mit dem Trigger in SQL können Sie einen Codestapel ausführen, wenn eine Einfügung, Aktualisierung oder eine andere Abfrage für eine bestimmte Tabelle ausgeführt wird.

Q27: What are the **different operators** available in SQL?

A27: There are three operators available in SQL, namely:

F27: Welche **verschiedenen Operatoren** sind in SQL verfügbar?

A27: In SQL stehen drei Operatoren zur Verfügung:

S27: SQL'de bulunan **farklı operatörler** nelerdir?

C27: SQL'de kullanılabilen üç operatör vardır:

Arithmetic Operators -	Arithmetische Operatoren -	Aritmetik Operatörler
Logical Operators -	Logische Operatoren -	Mantıksal Operatörler
Comparison Operators –	Vergleichsoperatoren -	Karşılaştırma Operatörleri

Q28: Are **NULL values** same as that of **zero** or a **blank space**?

A28: A NULL value is not at all same as that of zero or a blank space. NULL value represents a value which is unavailable, unknown, assigned or not applicable whereas a zero is a number and blank space is a character.

F28: Sind **NULL-Value** gleich **Null** oder **Leerzeichen**?

A28: Ein NULL-Value ist überhaupt nicht gleich dem von Null oder einem Leerzeichen. Der NULL-Value stellt einen Wert dar, der nicht verfügbar, unbekannt, zugewiesen oder nicht anwendbar ist, während eine Null eine Zahl und ein Leerzeichen ein Zeichen ist.

S28: **NULL değerleri sıfır** veya **boşlukla** aynı mı?

C28: NULL değeri, sıfır veya boşluk ile aynı değildir. NULL değeri, mevcut olmayan, bilinmeyen, atanan veya uygulanamayan bir değeri temsil ederken, sıfır bir sayı ve boşluk bir karakterdir.

Q29: What is the difference between **cross join** and **natural join**?

A29: The cross join produces the cross product or Cartesian product of two tables whereas the natural join is based on all the columns having the same name and data types in both the tables.

F29: Was ist der Unterschied zwischen **Cross Join** und **Natural Join**?

A29: Der Cross-Join erzeugt das Kreuzprodukt oder das kartesische Produkt zweier Tabellen, während der Natural-Join auf allen Spalten basiert, die in beiden Tabellen denselben Namen und denselben Datentyp haben.

S29: **Cross Join** (çapraz birleştirme) ve **Natural Join** (doğal birleştirme) arasındaki fark nedir?

C29: Çapraz birleştirme iki tablonun çapraz çarpımını veya Kartezyen çarpımını üretirken, doğal birleştirme her iki tablodaki aynı ada ve veri türlerine sahip tüm sütunlara dayanır.

Q30: What is **subquery** in SQL?

A30: In a subquery, the outer query is called as the main query whereas the inner query is called subquery. Subqueries are always executed first and the result of the subquery is passed on to the main query. It can be nested inside a SELECT, UPDATE or any other query. A subquery can also use any comparison operators such as >, < or =.

Q31: What are the different **types of a subquery**?

A31: There are two types of subquery namely, **Correlated** and **Non-Correlated**.

Q32: Explain them?

A32: **Correlated subquery**: These are queries which select the data from a table referenced in the outer query. It is not considered as an independent query as it refers to another table and refers the column in a table.

Non-Correlated subquery: This query is an independent query where the output of subquery is substituted in the main query.

F30: Was ist ein **Query** (eine Unterabfrage) in SQL?

A30: In einer Subquery wird die äußere Query als Hauptabfrage bezeichnet, während die innere Query als Subquery bezeichnet wird. Subquery werden immer zuerst ausgeführt und das Ergebnis der Subquery wird an die Hauptabfrage weitergeleitet. Es kann in einem SELECT, UPDATE oder einer anderen Abfrage verschachtelt sein. Eine Subquery kann auch beliebige Vergleichsoperatoren wie>, <oder = verwenden.

F31-32: Was sind die verschiedenen Arten eines **Subquery**? Und erklären Sie?

A31-32: **Correlated Subquery**: Dies sind Queries, die die Daten aus einer Tabelle auswählen, auf die in der äußeren Query verwiesen wird. Es wird nicht als unabhängige Abfrage betrachtet, da es auf eine andere Tabelle verweist und auf die Spalte in einer Tabelle verweist.

Non-Correlated Subquery: Diese Query ist eine unabhängige Query, bei der die Ausgabe von Subquery in der Hauptabfrage ersetzt wird.

S30: Subquery (alt sorgu) nedir?

C30: Bir Subquery ’de, dış sorgu ana sorgu olarak adlandırılırken, iç sorgu alt sorgu olarak adlandırılır. Alt sorgular her zaman önce yürütülür ve alt sorgunun sonucu ana sorguya aktarılır. Bir SELECT, UPDATE veya başka bir sorgunun içine yerleştirilebilir. Bir alt sorgu >, < veya = gibi herhangi bir karşılaştırma operatörünü de kullanabilir.

S31-32: Subquery çeşitleri nelerdir? Bunları açıklayınız?

C31-32: **Correlated Subquery** (İlişkili alt sorgu): Bunlar, dış sorguda referans verilen tablodan verileri seçen sorgulardır. Başka bir tabloya başvurduğu ve bir tablodaki sütuna başvurduğu için bağımsız bir sorgu olarak değerlendirilmez.

Non-Correlated Subquery (İlişkilendirilmemiş alt sorgu): Bu sorgu, alt sorgu çıktısının ana sorguda yer değiştirdiği bağımsız bir sorgudur.

Q33: List the ways to get the **count of records** in a table?

A33: To count the number of records in a table, you can use the below commands:

- SELECT COUNT(*) FROM table1
- SELECT rows FROM sysindexes WHERE id = OBJECT_ID(table1) AND indid < 2

Q34: Write a SQL query to find the names of employees that begin with **'A'**?

A34 To display name of the employees that begin with 'A', type in the below command:

- SELECT * FROM table_name WHERE emp_name like 'A%'

Q35: Write a SQL query to get the third highest salary of an employee from employee_table?

A35:

- SELECT TOP 1 salary FROM (SELECT TOP 3 salary FROM employee_table ORDER BY salary DESC) AS emp ORDER BY salary ASC;

(Not: **33**, **34** ve **35**. Sorular; Kodlar değişmeyeceği için Almanca ve Türkçesi yazılmamıştır.)

Q36: What is the **need for group functions** in SQL?

A36: Group functions work on the set of rows and returns one result per group. Some of the commonly used group functions are: **AVG, COUNT, MAX, MIN, SUM, VARIANCE**.

F36: Was sind **Gruppenfunktionen** in SQL **erforderlich**?

A36: Gruppenfunktionen arbeiten mit dem Satz von Zeilen und geben ein Ergebnis pro Gruppe zurück. Einige der häufig verwendeten Gruppenfunktionen sind: AVG, COUNT, MAX, MIN, SUM, VARIANCE.

S36: SQL'de **grup işlevleri için neler gereklidir**?

C36: Grup işlevleri satır kümesi üzerinde çalışır ve grup başına bir sonuç döndürür. Yaygın olarak kullanılan grup işlevlerinden bazıları şunlardır: AVG, COUNT, MAX, MIN, SUM, VARIANCE.

Q37: What is a **Relationship** and what are they?

F37: Was ist **Relationship** und welche Arten haben sie?

S37: Relationship nedir ve çeşitleri nelerdir?

A37: Relationships are defined as the **connection between the tables** in a database.

A37: Relationships sind definiert als die Verbindung zwischen den Tabellen in einer Datenbank.

C37: **Relationships** bir databankta tablolar arasındaki bağlantı olarak tanımlanır.

One to One Relationship.
One to Many Relationship.
Many to One Relationship.
Self-Referencing Relationship.

Q38: How can you insert **NULL values** in a column while inserting the data?

A38: Implicitly by omitting column from column list.

Explicitly by specifying NULL keyword in the VALUES clause.

F38: Wie können Sie beim Einfügen der Daten NULL-Values in eine Spalte einfügen?

Q38: Implizit durch Weglassen der Spalte in der Spaltenliste.

Explizit durch Angabe des NULL-Keywords in der VALUES-Klausel.

S38: Verileri eklerken bir sütuna, NULL değerleri nasıl ekleyebilirsin?

C38: Özel olarak sütun listesinden sütunu çıkararak.

Açıkça VALUES yan tümcesinde NULL anahtar kelime belirterek.

Q39: What is the main difference between '**BETWEEN'** and '**IN'** condition operators?

A39: BETWEEN-condition operator is used to display rows based on a range of values in a row whereas the IN-condition operator is used to check for values contained in a specific set of values.

Q40: Give an example please?

A40:

Example of **BETWEEN**:

SELECT * FROM students WHERE roll_no BETWEEN 10 AND 50;

Example of **IN**:

SELECT * FROM students WHERE roll_no IN (8, 15, 25);

F39: Was ist der Hauptunterschied zwischen den Condition-Operatoren " BETWEEN" und "IN"?

A39: Der BETWEEN-Condition-Operatoren wird verwendet, um Zeilen basierend auf einem Wertebereich in einer Zeile anzuzeigen, während der IN- Condition-Operatoren verwendet wird, um nach Werten zu suchen, die in einem bestimmten Wertesatz enthalten sind.

F40: Geben Sie bitte ein Beispiel?

A40:

Beispiel **BETWEEN**:

SELECT * FROM students WHERE roll_no BETWEEN 10 AND 50;

Beispiel **IN**:

SELECT * FROM students WHERE roll_no IN (8, 15, 25);

S39: "**BETWEEN**" ve "**IN**" koşulu operatörleri arasındaki temel fark nedir?

C39: BETWEEN operatörü, bir satırdaki bir değer aralığına göre satırları görüntülemek için kullanılırken, IN koşulu operatörü, belirli bir değer kümesinde bulunan değerleri kontrol etmek için kullanılır.

S40: Lütfen örnek veriniz?

C40:

Örneğin **BETWEEN**:

SELECT * FROM students WHERE roll_no BETWEEN 10 AND 50;

Örneğin **IN**:

SELECT * FROM students WHERE roll_no IN (8, 15, 25);

Q41: Why are **SQL functions** used?

A41:

- To perform some calculations on the data
- To modify individual data items
- To manipulate the output
- To format dates and numbers
- To convert the data types

F41: Warum werden SQL-Funktionen verwendet?

Q41:

- Um einige Berechnungen an den Daten durchzuführen.
- Um einzelne Datenelemente zu ändern.
- Um die Ausgabe zu manipulieren.
- Zum Formatieren von Datums- und Zahlenangaben.
- Um die Datentypen zu konvertieren.

S41: **SQL fonksiyonları** neden kullanılır?

C41:

- Veri üzerinde bazı hesaplamalar yapmak için.
- Bireysel veri öğelerini değiştirmek için.
- Çıktıyı değiştirmek için.
- Tarihleri ve sayıları biçimlendirmek için.
- Veri türlerini dönüştürmek için.

Q42: What is the need of **MERGE** statement?

A42: This statement allows conditional update or insertion of data into a table. It performs an **UPDATE** if a row exists, or an **INSERT** if the row does not exist.

F42: Was ist die Notwendigkeit ein MERGE-Statement?

A42: Diese Anweisung ermöglicht die bedingte Aktualisierung oder Einfügung von Daten in eine Tabelle. Es führt ein **UPDATE** durch, wenn eine Zeile vorhanden ist, oder ein **INSERT**, wenn die Zeile nicht vorhanden ist.

S42: MERGE ifadesine neden ihtiyaç duyulur?

C42: Bu ifade, koşullu güncellemeye veya bir tabloya veri eklenmesine izin verir. Bir satır varsa UPDATE güncelleme veya satır yoksa INSERT ekleme gerçekleştirir.

Q43: What do you mean by **recursive stored procedure**?

A43: Recursive stored procedure refers to a stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps the programmers to use the same set of code n number of times.

F43: Was meinen Sie mit **Recursive stored Prozedur**?

A43: Recursive stored procedure refers to a stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps the programmers to use the same set of code n number of times.

S43: **Recursive stored prosedür** ile ne demek istiyorsun?

C43: **Recursive stored prosedürü**, bazı sınır koşullarına ulaşana kadar kendi kendine çağıran saklı bir prosedürü ifade eder. Bu özyinelemeli fonksiyon veya prosedür, programcıların aynı kod setini n sayıda kullanmasına yardımcı olur.

Q44: What is **Clause** in SQL?

A44: SQL clause helps to limit the result set by providing a condition to the query. A clause helps to filter the rows from the entire set of records. For example – WHERE, HAVING clause.

Q45: What is the difference between '**HAVING'** CLAUSE and a '**WHERE'** CLAUSE?

A45:

HAVING clause can be used only with SELECT statement. It is usually used in a GROUP BY clause and whenever GROUP BY is not used, HAVING behaves like a WHERE clause.

HAVING Clause is only used with the GROUP BY function in a query whereas WHERE Clause is applied to each row before they are a part of the GROUP BY function in a query.

F44: Was ist **Clause** in SQL?

A44: Die SQL-Klausel hilft, die Ergebnismenge zu begrenzen, indem eine Bedingung für die Abfrage bereitgestellt wird. Eine Klausel hilft dabei, die Zeilen aus dem gesamten Datensatzsatz zu filtern. Zum Beispiel - WHERE, HAVING-Klausel.

F45: Was ist der Unterschied zwischen "**HAVING**" - Clause und "**WHERE**" - Clause?

A45:

Die HAVING-Klausel kann nur mit der SELECT-Anweisung verwendet werden. Es wird normalerweise in einer GROUP BY-Klausel verwendet. Wenn GROUP BY nicht verwendet wird, verhält sich HAVING wie eine WHERE-Klausel.

Die HAVING-Klausel wird nur mit der GROUP BY-Funktion in einer Abfrage verwendet, während die WHERE-Klausel auf jede Zeile angewendet wird, bevor sie Teil der GROUP BY-Funktion in einer Abfrage ist.

S44: SQL de **Clause** nedir?

C44: SQL Clause, sorguya bir koşul sağlayarak sonuç kümesini sınırlamaya yardımcı olur. Clause, tüm kayıt kümesindeki satırları filtrelemeye yardımcı olur. Örneğin WHERE, HAVING Clause.

S45: 'HAVING' CLAUSE ve 'WHERE' CLAUSE arasındaki fark nedir?

C45:

HAVING tümcesi yalnızca SELECT deyimi ile kullanılabilir. Genellikle bir GROUP BY yan tümcesinde kullanılır ve GROUP BY kullanılmadığında HAVING bir WHERE tümcesi gibi davranır.

HAVING Tümce, bir sorguda yalnızca GROUP BY işlevi ile kullanılırken, WHERE İfadesi, bir sorguda GROUP BY işlevinin parçası olmadan önce her satıra uygulanır.

Q46: List the ways in which **Dynamic SQL** can be executed?

A46: Write a query with parameters. Using EXEC. Using sp_executesql.

F46: Auflisten, wie **Dynamic SQL** ausgeführt werden kann?

A46: Schreiben Sie eine Abfrage mit Parametern. Verwenden von EXEC. Verwenden von sp_executesql.

S46**: Dinamik SQL**'in çalıştırılabileceği yolları listeleyin.

C46: Parametrelerle bir query sorgu yazın. EXEC kullanın. Sp_executesql kullanın.

Q47: What are the various levels of **constraints**?

A47: Constraints are the representation of a column to enforce data entity and consistency. There are two levels of a constraint, namely: column level constraint, table level constraint.

F47: Was sind die verschiedenen **Constraints** (Einschränkungen)?

A47: Constraints sind die Darstellung einer Spalte, um die Datenentität und -konsistenz zu erzwingen. Es gibt zwei Ebenen einer Constraints, nämlich: Constraints auf Spaltenebene (column level), Constraints auf Tabellenebene (table level).

S47: Çeşitli kısıtlama **Constraints** seviyeleri nelerdir?

C47: Constraints (Kısıtlamalar), veri varlığını ve tutarlılığını zorlamak için bir sütunun temsilidir. Bir kısıtlamanın iki düzeyi vardır: sütun düzeyinde kısıtlama, tablo düzeyinde kısıtlama.

Q48: How can you fetch common records from **two tables**?

A48: You can fetch common records from two tables using **INTERSECT**.

Q49: Give an example please?

A49:

SELECT studentID FROM student **INTERSECT** SELECT studentID FROM Exam

F48: Wie können Sie **gemeinsame Datensätze** aus zwei Tabellen abrufen?

A48: Mit **INTERSECT**. Können Sie allgemeine Datensätze aus zwei Tabellen abrufen.

F49: Geben Sie bitte ein Beispiel?

A49:

SELECT studentID FROM student **INTERSECT** SELECT studentID FROM Exam

S48: Ortak kayıtları iki tablodan nasıl getirebilirsiniz?

C48: INTERSECT kullanarak ortak kayıtları iki tablodan alabilirsiniz.

S49: Lütfen bir örnek veriniz?

A49:

SELECT studentID FROM student **INTERSECT** SELECT studentID FROM Exam

Q50: List some **case manipulation functions** in SQL?

A50: **LOWER**: This function returns the string in lowercase. It takes a string as an argument and returns it by converting it into lower case.

Syntax: LOWER('string')

UPPER: This function returns the string in uppercase. It takes a string as an argument and returns it by converting it into uppercase.

Syntax: UPPER('string')

INITCAP: This function returns the string with the first letter in uppercase and rest of the letters in lowercase.

Syntax: INITCAP('string')

F50: Einige Fallmanipulationsfunktionen (**Case Manipulation Funktions**) in SQL auflisten?

A50: **LOWER**: Diese Funktion gibt die Zeichenfolge in Kleinbuchstaben zurück. Es nimmt eine Zeichenfolge als Argument und gibt sie zurück, indem sie in Kleinbuchstaben konvertiert wird.

Syntax: LOWER('string')

UPPER: Diese Funktion gibt die Zeichenfolge in Großbuchstaben zurück. Es nimmt eine Zeichenfolge als Argument und gibt sie zurück, indem sie in Großbuchstaben konvertiert wird.

Syntax: UPPER('string')

INITCAP: Diese Funktion gibt die Zeichenfolge mit dem ersten Buchstaben in Großbuchstaben und den restlichen Buchstaben in Kleinbuchstaben zurück.

Syntax: INITCAP('string')

S50: SQL'deki bazı durum işleme işlevlerini (**case manipulation functions**) listeler misiniz?

C50: **LOWER**: Bu işlev dizeyi küçük harfle döndürür. Bir dizgeyi bağımsız değişken olarak alır ve onu küçük harfe dönüştürerek döndürür.

Syntax: LOWER('string')

UPPER: Bu işlev dizeyi büyük harfle döndürür. Bir dizgeyi argüman olarak alır ve onu büyük harfe dönüştürerek döndürür.

Syntax: UPPER('string')

INITCAP: Bu işlev dizeyi ilk harfi büyük, geri kalanı küçük harfle döndürür.

Syntax: INITCAP('string')

Q51: What are the different **set operators available** in SQL?

A51: Some of the available set operators are – **Union**, **Intersect** or **Minus operators.**

F51: Welche verschiedenen Mengenoperatoren sind in SQL verfügbar?

A51: Einige der verfügbaren Mengenoperatoren sind - **Union**-, **Intersect**- oder **Minus**-**Operatoren**.

S51: SQL'de bulunan farklı küme operatörleri nelerdir?

C51: Kullanılabilir küme operatörlerinden bazıları => **Union**, **Intersect** veya **Minus** operators.

Q52: What is an **ALIAS** command?

A52: ALIAS name can be given to any table or a column. This alias name can be referred in WHERE clause to identify a particular table or a column.

F52: Was ist ein **ALIAS**-Command?

A52: Der ALIAS-Name kann jeder Tabelle oder Spalte zugewiesen werden. Auf diesen Aliasnamen kann in der WHERE-Klausel verwiesen werden, um eine bestimmte Tabelle oder Spalte zu identifizieren.

S52: **ALIAS** komutu nedir?

C52: ALIAS adı herhangi bir tabloya veya bir sütuna verilebilir. Bu takma ad, belirli bir tabloyu veya bir sütunu tanımlamak için WHERE yan tümcesinde belirtilebilir.

Q53: What is **aggregate** function?

A53: Aggregate functions are used to evaluate mathematical calculation and returns a single value. These calculations are done from the columns in a table. For example- **max(), count()** are calculated with respect to numeric.

F53: Was ist eine **Aggregatfunktion**?

A53: Aggregatfunktionen werden zur Auswertung der mathematischen Berechnung verwendet und geben einen einzelnen Wert zurück. Diese Berechnungen werden aus den Spalten in einer Tabelle durchgeführt. Zum Beispiel werden **max (), count ()** in Bezug auf numerisch berechnet.

S53: **Aggregate** fonksiyonu nedir?

C53: Aggregate fonksiyonu; Toplama işlevleri, matematiksel hesaplamayı değerlendirmek için kullanılır ve tek bir değer döndürür. Bu hesaplamalar bir tablodaki sütunlardan yapılır. Örneğin, **max (), count ()** sayısal olarak hesaplanır.

Q54: What is **scalar function**?

A54: Scalar functions return a single value based on the input value. For example – **UCASE(), NOW()** are calculated with respect to string.

F54: Was ist Skalar-Funktion?

A54: Skalar-Funktionen geben einen einzelnen Wert basierend auf dem Eingabewert zurück. Zum Beispiel - **UCASE (), NOW ()** werden in Bezug auf die Zeichenfolge berechnet.

S54: **Scalar fonksiyon** nedir?

C54: Skaler işlevler, giriş değerine göre tek bir değer döndürür. Örneğin- **UCASE (), NOW ()** dizeye göre hesaplanır.

Q55: How can you **fetch alternate records** from a table?

A55: You can fetch alternate records i.e. both odd and even row numbers. For example: To display even numbers, use the following command:

SELECT studentId FROM (SELECT rowno, studentId FROM student) WHERE mod(rowno,2)=0

F55: Wie können Sie alternative Datensätze aus einer Tabelle abrufen?

A55: Sie können alternative Datensätze abrufen, d. H. Sowohl ungerade als auch gerade Zeilennummern. Beispiel: Verwenden Sie den folgenden Befehl, um gerade Zahlen anzuzeigen:

SELECT studentId FROM (SELECT rowno, studentId FROM student) WHERE mod(rowno,2)=0

S55: Bir tablodan alternatif kayıtları nasıl getirebilirsiniz?

C55: Alternatif kayıtları, yani hem tek hem de çift satır numaralarını getirebilirsiniz. Örneğin; Çift sayıları görüntülemek için aşağıdaki komutu kullanın:

SELECT studentId FROM (SELECT rowno, studentId FROM student) WHERE mod(rowno,2)=0

Q56: How can you select **unique records** from a table?

A56: You can select unique records from a table by using the **DISTINCT** keyword.

```
SELECT DISTINCT studentID FROM Student;
```

F56: Wie können Sie **eindeutige Datensätze** aus einer Tabelle auswählen?

A56: Mit dem Schlüsselwort DISTINCT können Sie eindeutige Datensätze aus einer Tabelle auswählen.

```
SELECT DISTINCT studentID FROM Student;
```

S56: Bir tablodan benzersiz kayıtları nasıl seçebilirsiniz?

C56: DISTINCT anahtar sözcüğünü kullanarak bir tablodan benzersiz kayıtlar seçebilirsiniz.

```
SELECT DISTINCT studentID FROM Student;
```

Q57: How can you fetch **first 5 characters** of the string?

A57: SELECT SUBSTRING(StudentName, 1, 5) AS studentname FROM student

F57: Wie können Sie die **ersten 5 Zeichen** der Zeichenfolge abrufen?

A57: SELECT SUBSTRING(StudentName, 1, 5) AS studentname FROM student

S57: Dizenin ilk 5 karakterini nasıl getirebilirsiniz?

c57: SELECT SUBSTRING(StudentName, 1, 5) AS studentname FROM student

Q58: What is the main difference between **SQL** and **PL/SQL**?

A58:

SQL is a query language that allows you to issue a single query or execute a single INSERT/UPDATE/DELETE whereas;

PL/SQL is Oracle's "Procedural Language" SQL, which allows you to write a full program (loops, variables, etc.) to accomplish multiple operations such as SELECTS/INSERTS/UPDATES/DELETES.

F58: Was ist der Hauptunterschied zwischen **SQL** und **PL / SQL**?

A58:

SQL ist eine Abfragesprache, mit der Sie eine einzelne Abfrage ausgeben oder eine einzelne INSERT / UPDATE / DELETE ausführen können,

während PL / SQL die SQL-Datei "Procedural Language" von Oracle ist, mit der Sie ein vollständiges Programm (Schleifen, Variablen usw.) Schreiben können .) um mehrere Vorgänge wie SELECTS / INSERTS / UPDATES / DELETES auszuführen.

S58: **SQL** ile **PL/SQL** arasındaki fark nedir?

C58:

SQL, tek bir sorgu yayınlamanıza veya tek bir INSERT / UPDATE / DELETE işlemi gerçekleştirmenize izin veren bir sorgu dilidir;

PL / SQL ise Oracle'ın "Prosedür Dili" SQLidir ve tam bir program (döngüler, değişkenler vb.) SELECTS / INSERTS / UPDATES / DELETES gibi birden çok işlemi gerçekleştirmek için.

Q59: What is a **View**?

A59: A view is a virtual table which consists of a subset of data contained in a table. Since views are not present, it takes less space to store. View can have data of one or more tables combined and it depends on the relationship.

F59: Was ist eine Ansicht (**View**)?

A59: Eine (View) Ansicht ist eine virtuelle Tabelle, die aus einer Teilmenge von Daten besteht, die in einer Tabelle enthalten sind. Da Ansichten nicht vorhanden sind, wird weniger Speicherplatz benötigt. In der Ansicht können Daten einer oder mehrerer Tabellen kombiniert werden. Dies hängt von der Beziehung ab.

S59: View nedir?

C59: (View) Görünüm, bir tabloda yer alan verilerin bir alt kümesinden oluşan sanal bir tablodur. Görünümler olmadığından, depolanması daha az yer kaplar. Görünüm, bir veya daha fazla tablonun verilerini birleştirebilir ve bu, ilişkiye bağlıdır.

Q60: What is a **Stored Procedure**?

A60: A Stored Procedure is a function which consists of many SQL statements to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required which saves time and avoid writing code again and again.

Q61: List some **advantages** and **disadvantages** of Stored Procedure?

A61:

Advantages: A Stored Procedure can be used as a modular programming which means create once, store and call for several times whenever it is required. This supports faster execution.

Disadvantage: The only disadvantage of Stored Procedure is that it can be executed only in the database and utilizes more memory in the database server.

F60: Was ist eine gespeicherte Prozedur (**Stored Procedure**)?

A60: Eine gespeicherte Prozedur ist eine Funktion, die aus vielen SQL-Anweisungen für den Zugriff auf das Datenbanksystem besteht. Mehrere SQL-Anweisungen werden in einer gespeicherten Prozedur konsolidiert und wann und wo immer erforderlich ausgeführt. Dies spart Zeit und vermeidet das wiederholte Schreiben von Code.

F61: Einige **Vor**- und **Nachteile** der **Stored Procedure** auflisten?

A61: Vorteile: Eine gespeicherte Prozedur kann als modulare Programmierung verwendet werden, d.h. einmal erstellen, speichern und mehrmals aufrufen, wann immer dies erforderlich ist. Dies unterstützt eine schnellere Ausführung.

Nachteil: Der einzige Nachteil der gespeicherten Prozedur besteht darin, dass sie nur in der Datenbank ausgeführt werden kann und mehr Speicher auf dem Datenbankserver belegt.

S60: Depolanan Prosedür (**Stored Procedure**) nedir?

C60: Depolanan Prosedür (**Stored Procedure**), veritabanı sistemine erişmek için birçok SQL deyiminden oluşan bir işlevdir. Birkaç SQL deyimi bir saklı yordamda birleştirilir ve gerektiğinde bunları her zaman ve yerde yürütür, bu da zamandan tasarruf sağlar ve tekrar tekrar kod yazmaktan kaçınır.

S61: Depolanan Prosedürün bazı **avantaj** ve **dezavantajlarını** listeleyin.

C61: Avantajlar: Depolanan Prosedür, modüler bir programlama olarak kullanılabilir; bu, gerektiğinde bir kez oluşturma, kaydetme ve birkaç kez çağırma anlamına gelir. Bu, daha hızlı yürütmeyi destekler.

Dezavantaj: Depolanan Prosedürün tek dezavantajı, yalnızca veritabanında çalıştırılabilmesi ve veritabanı sunucusunda daha fazla bellek kullanmasıdır.

Q62: What do you mean by **Collation**?

A62: Collation is defined as a set of rules that determine how data can be sorted as well as compared.

Q63: What are the different **types of Collation Sensitivity**?

A63:

Case Sensitivity: A and a and B and b.

Kana Sensitivity: Japanese Kana characters.

Width Sensitivity: Single byte character and double-byte character.

Accent Sensitivity:

F62: Was meinst du mit **Kollation**?

A62: Die Kollation ist definiert als eine Reihe von Regeln, die bestimmen, wie Daten sortiert und verglichen werden können.

F63: Was sind die verschiedenen Arten der **Kollation-Sensitivity**?

A63:

Case Sensitivity: A und a und B und b.

Kana Sensitivity: Japanische Kana-Zeichen.

Width Sensitivity: Einzelbytezeichen und Doppelbytezeichen.

Accent Sensitivity:

S62: **Collation** (Harmanlama) ile ne demek istiyorsun?

C62: Collation verilerin nasıl sıralanacağını ve karşılaştırılacağını belirleyen bir dizi kural olarak tanımlanır

S63: Collation Sensitivity (Harmanlama Duyarlılığı)'nin farklı türleri nelerdir?

C63:

Case Sensitivity (Büyük / Küçük Harfe Duyarlılık): A ve a ve B ve b.

Kana Hassasiyeti (Kana Sensitivity): Japon Kana karakterleri.

Genişlik Hassasiyeti (Width Sensitivity): Tek baytlık karakter ve çift baytlık karakter.

Vurgu Hassasiyeti (Accent Sensitivity).

Q64: What is **Auto Increment** in SQL?

A64: Auto increment keyword allows the user to create a unique number to get generated whenever a new record is inserted into the table. This keyword is usually required whenever PRIMARY KEY is used.

F64: Was ist **Auto Inkrement** in SQL?

A64: Mit dem Schlüsselwort "Automatisches Inkrementieren" kann der Benutzer eine eindeutige Nummer erstellen, die generiert wird, wenn ein neuer Datensatz in die Tabelle eingefügt wird. Dieses Schlüsselwort wird normalerweise benötigt, wenn PRIMARY KEY verwendet wird.

S64: SQL'de Otomatik Artış (**Auto Inkrement**) nedir?

C64: Otomatik artış anahtar sözcüğü, kullanıcının tabloya her yeni kayıt eklendiğinde oluşturulması için benzersiz bir numara oluşturmasına olanak tanır. Bu anahtar kelime genellikle PRIMARY KEY her kullanıldığında gereklidir.

Q65: What is a **Datawarehouse**?

A65: Datawarehouse refers to a central repository of data where the data is assembled from multiple sources of information. Those data are consolidated, transformed and made available for the mining as well as online processing.

F65: Was ist ein Datawarehouse?

A65: Datawarehouse bezieht sich auf ein zentrales Datenrepository, in dem die Daten aus mehreren Informationsquellen zusammengestellt werden. Diese Daten werden konsolidiert, transformiert und für den Bergbau sowie die Online-Verarbeitung zur Verfügung gestellt.

S65: Datawarehouse nedir?

C65: Datawarehouse, verilerin birden fazla bilgi kaynağından toplandığı merkezi bir veri havuzunu ifade eder. Bu veriler konsolide edilir, dönüştürülür ve madencilik ve çevrimiçi işleme için kullanılabilir hale getirilir.

Q66: What is **STUFF** and **REPLACE function**?

A66:

STUFF function: This function is used to overwrite existing character or inserts a string into another string.

Syntax: STUFF(string_expression,start, length, replacement_characters).

REPLACE function: This function is used to replace the existing characters of all the occurrences.

Syntax: REPLACE(string_expression, search_string, replacement_string)

F66: Was sind die **Funktionen STUFF** und **REPLACE**?

A66:

STUFF-Funktion: Mit dieser Funktion werden vorhandene Zeichen überschrieben oder eine Zeichenfolge in eine andere Zeichenfolge eingefügt.

Syntax: STUFF(string_expression,start, length, replacement_characters).

REPLACE-Funktion: Diese Funktion wird verwendet, um die vorhandenen Zeichen aller Vorkommen zu ersetzen.

Syntax: REPLACE(string_expression, search_string, replacement_string)

S66: **STUFF** ve **REPLACE fonksiyon** nedir?

C66:

STUFF Fonksiyonu: Bu fonksiyon, mevcut karakterin üzerine yazmak veya başka bir dizgeye bir dizi eklemek için kullanılır.

Sözdizimi: STUFF (dizgi_ifadesi, başlangıç, uzunluk, değiştirme_ karakterleri).

STUFF(string_expression,start, length, replacement_characters).

REPLACE işlevi: Bu işlev, tüm oluşumların mevcut karakterlerini değiştirmek için kullanılır.

Sözdizimi: DEĞİŞTİR (dize_ifadesi, arama_dizesi, değiştirme_dizesi)

REPLACE(string_expression, search_string, replacement_string)