Question Code T3A1.2Q1

***Describe** the properties of a database table in a relational database.*

To describe the properties of a relational database table, the following example is used.

```
CREATE TABLE Table1
(
    TableId INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    TableTitle VARCHAR (160) NOT NULL,
    SomeTableId INTEGER NOT NULL REFERENCES SomeTable
);

CREATE INDEX IFK_SomeTableId
    ON SomeTable (SomeTableId);
```

1. **Columns**: Each table has several columns, which is a vertical set of cells with data values of the same type. In the above example, TableId, TableTitle and ParentTableId are all columns. If a single column is used as PRIMARY KEY, then the column cannot have duplicate values. There are COMPOSITE PRIMARY KEY, where multiple columns form part of the primary key, where the combination of values in the columns in a row should be unique. A column can be restricted to only having values from a column in another table, and the key to link these two tables is called FOREIGN KEY. Primary key values can be auto entered with every insert using sequences. There can other constraints like NOT NULL constraint, which forces the column to always have a value, UNIQUE constraint to have unique values and check constraints to add data validation.
2. **Rows**: A row is a horizontal set of cells with a data value from each column. Each row has a unique row id that is implemented in the database. As row id is internal to the database, primary key columns should be used to uniquely identify a record. A row is also called the record.

Question Code T3A1.2Q2

***Identify** the three possible relationships between entities in a relational database and **explain** how primary and foreign keys are used to create a many-to-many relationship between tables.*

The image below contains the three entity relationships in a relational database.

One-to-One

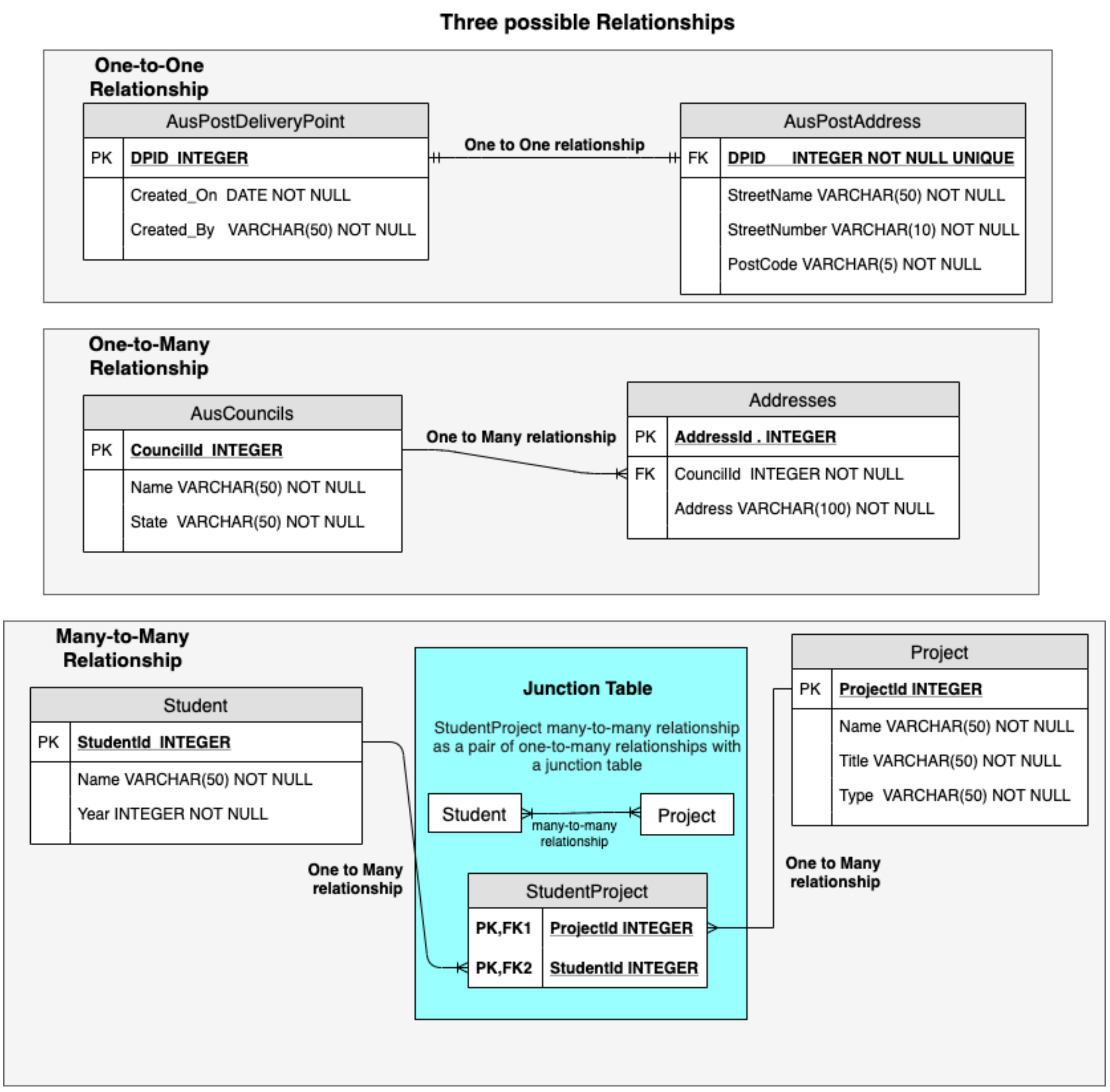
The relationship between AusPostDeliveryPoint and AusPostAddress is a one-to-one relationship. Every address in Australia has a delivery point id, and it is exactly mapped to one address. It helps with delivery of postal and other services. One-to-One relationship can be enforced using Primary Key and Foreign Key + Unique Constraint on the related table.

One-to-Many

The relationship between AusCouncils and Addresses entities is a one-to-many relationship. A field in one entity, is used in multiple records in the related entity. One-to-many relationships can be enforced by using the primary key of a table as the foreign key of the related table. One-to-Many relationships is very common.

Many-to-Many

In Many-to-Many relationships, a field in one entity will be in multiple records in the related entity and vice versa. In the entities listed below, a student can be in many projects and a project can be in many students. In relational databases, it is implemented using junction tables, which is a pair of one-to-many relationships. The junction table has many students and many projects.


Three possible Relationships
One-to-One Relationship
AusPostDeliveryPoint
PK DPID INTEGER
Created_On DATE NOT NULL
Created_By VARCHAR(50) NOT NULL
One to One relationship
AusPostAddress
FK DPID INTEGER NOT NULL UNIQUE
StreetName VARCHAR(50) NOT NULL
StreetNumber VARCHAR(10) NOT NULL
PostCode VARCHAR(5) NOT NULL
One-to-Many Relationship
AusCouncils
PK CouncilId INTEGER
Name VARCHAR(50) NOT NULL
State VARCHAR(50) NOT NULL
One to Many relationship
Addresses
PK AddressId . INTEGER
FK CouncilId INTEGER NOT NULL
Address VARCHAR(100) NOT NULL
Many-to-Many Relationship
Student
PK StudentId INTEGER
Name VARCHAR(50) NOT NULL
Year INTEGER NOT NULL
One to Many relationship
Junction Table
StudentProject many-to-many relationship as a pair of one-to-many relationships with a junction table
Student
many-to-many relationship
Project
StudentProject
PK,FK1 ProjectId INTEGER
PK,FK2 StudentId INTEGER
Project
PK ProjectId INTEGER
Name VARCHAR(50) NOT NULL
Title VARCHAR(50) NOT NULL
Type VARCHAR(50) NOT NULL
One to Many relationship

Use of Primary Key and Foreign Key in Many-to-Many Relationships

As mentioned above, many-to-many relationships are implemented using junction tables.

If we want to set up a many-to-many relationships between Student and Project, we get the primary keys of both tables. These primary keys are the foreign keys in the junction table.

```
CREATE TABLE Student (
    StudentId INTEGER PRIMARY KEY,
    Name VARCHAR (50) NOT NULL,
    Year INTEGER NOT NULL
);

CREATE TABLE Project (
    ProjectId INTEGER PRIMARY KEY,
    Title VARCHAR (50) NOT NULL,
    Type VARCHAR (50) NOT NULL
);
```

StudentId and ProjectId are the primary keys in Student and Project. They are the foreign keys in StudentProject table. We don't want to have duplicate student/project combination and also junction table requires a primary key. Instead of creating a new primary key column, composite primary key can be used combining the foreign keys. This constraint will also prevent the duplication of student/project combination.

```sql
CREATE TABLE StudentProject (
    ProjectId INTEGER,
    StudentId INTEGER,
    FOREIGN KEY (ProjectId) REFERENCES Project (ProjectId),
    FOREIGN KEY (StudentId) REFERENCES Student (StudentId),
    PRIMARY KEY (ProjectId, StudentId)
);
```

Question Code T3A1.2Q3

*Constraints are restrictions on data. **Describe** three constraints which can be applied to ensure data integrity in a relational database.*

The following are three of the constraints that can ensure data integrity.

Check constraint

Check constraint is used to make sure a certain condition is true before inserting data into table. The expression should to true to satisfy constraint. Otherwise the insert will cause check constraint violation. The following SQL uses check constraint to validate the length of UserId and validate the Age.

```
CREATE TABLE FaceBookAccount (
    UserId VARCHAR (50) NOT NULL,
    LastName VARCHAR (255) NOT NULL,
    FirstName VARCHAR (255),
    Age INTEGER,
    CONSTRAINT CHK_AGE CHECK (Age>=18),
    CONSTRAINT CHK_LEN CHECK (LENGTH(UserId) > 8)
);
```

Unique constraint

Unique constraint is used to make sure that the value in a column is unique. In the following SQL, SchoolId in SuburbSchool table is Unique and Foreign Key. Here, unique constraint is used to have a one-to-one relationship between School and SuburbSchool.

```
CREATE TABLE School (
    SchoolId INTEGER PRIMARY KEY,
    Name VARCHAR (50) NOT NULL,
    Type VARCHAR (50) NOT NULL
);

CREATE TABLE SuburbSchool (
    SchoolId INTEGER UNIQUE,
    SuburbName VARCHAR(20),
    FOREIGN KEY (SchoolId) REFERENCES School (SchoolId)
);
```

Not Null Constraint

Not Null constraint forces the column to have a value. In the following SQL, Not Null constraint is used to always have a title.

```
CREATE TABLE BOOK (
    BookId INTEGER PRIMARY KEY,
    Title VARCHAR (50) NOT NULL
);
```

Question Code T3A1.2Q4

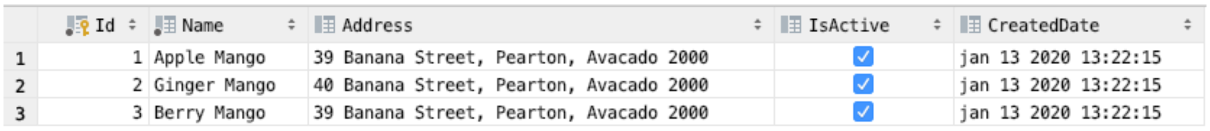
***Explain** how data types enforce data integrity in a relational database.*

There are three main data types in SQL. Depending on the data type, there are restrictions on the data that can be entered into the columns.

The following SQL is used to explain enforcement of data integrity using data types.

```
CREATE TABLE Customer
(
    Id INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Name VARCHAR (40) NOT NULL,
    Address VARCHAR (100),
    IsActive BOOLEAN,
    CreatedDate DATETIME
);

INSERT INTO Customer (Name, Address, IsActive, CreatedDate)
 VALUES ('Apple Mango', '39 Banana Street, Pearton, Avacado 2000',
        1, 'jan 13 2020 13:22:15');
```



	Id	Name	Address	IsActive	CreatedDate
1	1	Apple Mango	39 Banana Street, Pearton, Avacado 2000	✓	jan 13 2020 13:22:15
2	2	Ginger Mango	40 Banana Street, Pearton, Avacado 2000	✓	jan 13 2020 13:22:15
3	3	Berry Mango	39 Banana Street, Pearton, Avacado 2000	✓	jan 13 2020 13:22:15

Sample Error, if incorrect data type is used while insertion.

```
INSERT INTO Customer (Id, Name, Address, IsActive, CreatedDate)
 VALUES ('abcdef','Ginger Mango', '40 Banana Street, Pearton, Avacado 2000',
        1, 'jan 13 2020 13:22:15');
```

[2020-01-13 10:38:08] [20] [SQLITE_MISMATCH] Data type mismatch (datatype mismatch)

Numeric Data Type

In the above example Id is an Integer. Integer is a type of Numeric Data Type. It will prevent the inserting of characters into this field. Boolean data type is also used in the above example, it is also a numeric data type with 0 or 1 as the values representing True or False. There are other numeric sub data types supported by databases like Oracle and MySQL. These would allow to restrict the size and also restrict the number of decimal places.

Date and Time Data Type

In the above example, CreatedDate is of DATETIME data type. In databases, Date and Time data type column will restrict the values to be date or datetime depending on the data type used. The enforces the column to have a valid date and time.

String Data Type

In the above example Name and Address are of type VARCHAR, which is a subtype of String Data type. This column can contain numbers, characters and special characters. VARCHAR has a parameter size, which restricts the maximum length of column. To store large objects like images and documents, BLOB data type can be used. BLOB is a String Data subtype. The contents of the BLOB can only be validated by the application inserting data into BLOB columns. It is possible to restrict the maximum size of the object in the BLOB column by using the size parameter.

Question Code T3A1.2Q5

*SQL provides operations for the definition and manipulation of data in a relational database. **Describe** the function of three operators and support your answer with an example.*

Three operators used for definition and manipulation of data in relational database.

CREATE

CREATE is used to define table structure. It is one of commands in DDL (Data Definition Language)

All the data required to create a table can be included in CREATE command. It requires table name, column names and column types. Apart from these constraints can also be included in the CREATE command. The constraints used in the following SQL are PRIMARY KEY and NOT NULL. Other constraints that can be used are UNIQUE, FOREIGN KEY and Check. AUTOINCREMENT generates a unique number automatically when a record is inserted without Id.

```
CREATE TABLE Individual
(
    Id INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    Name VARCHAR (40) NOT NULL,
    Address VARCHAR (100)
);
```

It is also possible to create a table by copying structure and data from one or more other tables using SELECT queries.

```
CREATE TABLE IndividualCopy AS
SELECT Id, Name, Address
FROM Individual;
```

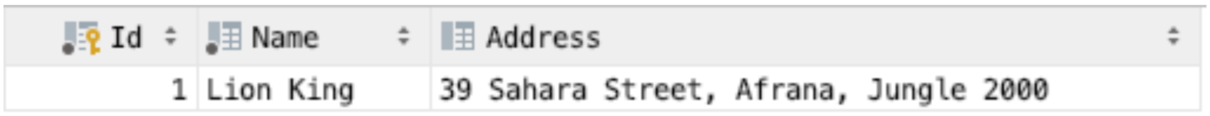
INSERT

INSERT is used to insert data into a table. It is one of the commands in DML (Data Manipulation Language)

INSERT command is used to insert a record into a table. INSERT should contain all the columns for which data values are provided. It contains columns followed by values. The values should match the data type of the column and all constraint violation should be avoided. If there are values for every column in the table in the insert statement, then the column names in the insert command are optional.

```
INSERT INTO Individual (Name, Address)
 VALUES ('Lion King', '39 Sahara Street, Afrana, Jungle 2000');
```

```sql
-- insert successful
SELECT * FROM Individual;
```



Id	Name	Address
1	Lion King	39 Sahara Street, Afrana, Jungle 2000

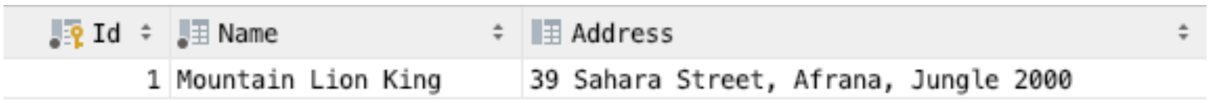
UPDATE

UPDATE is used to modify data that is in a table. It is one of the commands in DML (Data Manipulation Language)

UPDATE is used to modify one or more records in a table. An update statement contains the table name followed by SET to specify the new values for columns. It is followed by a WHERE clause to specify a condition to be true to identify and modify the record. Though UPDATE can work without a where clause, it is always better to use where clause to avoid updating unnecessary records.

```sql
UPDATE Individual SET Name = 'Mountain Lion King'
WHERE Id = 1;

-- update successful
SELECT * FROM Individual;
```



Id	Name	Address
1	Mountain Lion King	39 Sahara Street, Afrana, Jungle 2000

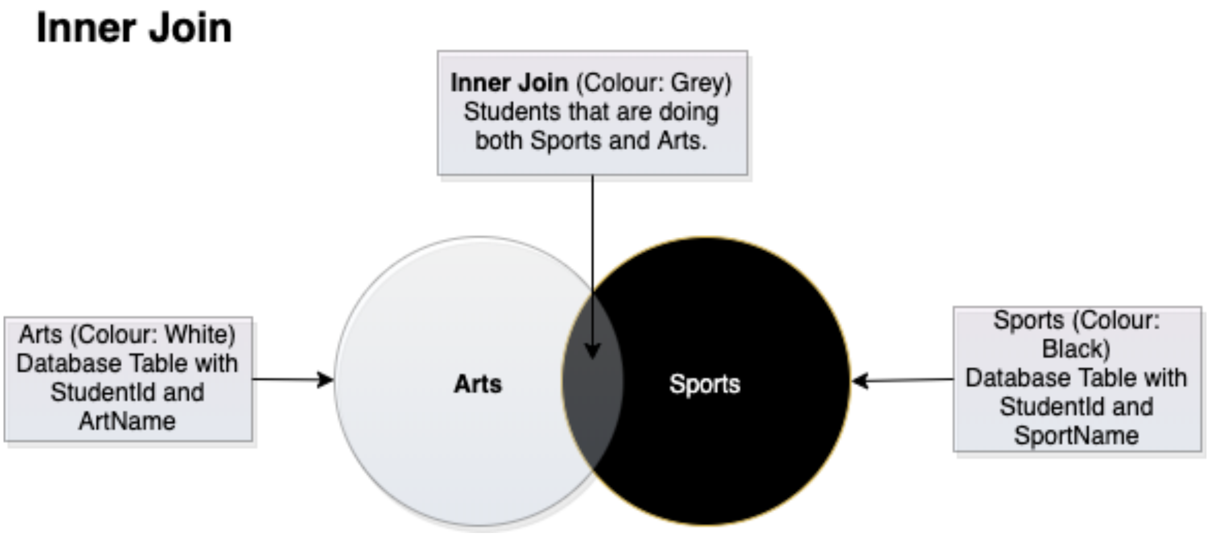
Question Code T3A1.2Q6

*Relational databases make data manipulation highly effective using SQL queries. **Describe** how the `INNER JOIN` query retrieves data from multiple tables.*

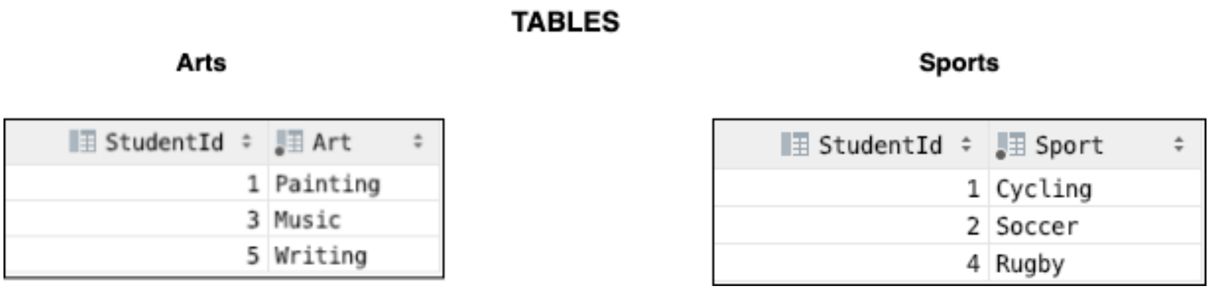
In relational databases, data required for applications or analysis is stored in multiple tables. It is not always necessary to create junction tables to link all this data together, as it may result in many unnecessary tables that is not often used. Joins help with retrieving the required data. One of the Joins that help with retrieving the data that has a common link between two tables with the linking value present in both tables is called INNER JOIN. Inner Join is the intersection of two sets of data. Inner Joins can be used in materialised views for data analytics of a large volume of data.

The following example describes how inner join works.

There are two sets of data named Arts and Sports. We would like to find the students that are in both sets of data. We are trying the get the data in the intersection of Sports and Arts.


Inner Join
Inner Join (Colour: Grey) Students that are doing both Sports and Arts.
Arts (Colour: White) Database Table with StudentId and ArtName
Arts
Sports
Sports (Colour: Black) Database Table with StudentId and SportName

In the database there are two tables Arts and Sports. We can see that the common column between Arts and Sports in StudentId. If we look at the data, not every student is registered for Arts and Sports. There is only one student with StudentId 1, who is in both tables. Inner Join will just retrieve that student, who is in both tables.



TABLES

Arts

StudentId	Art
1	Painting
3	Music
5	Writing

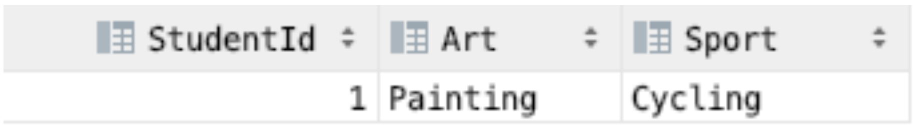
Sports

StudentId	Sport
1	Cycling
2	Soccer
4	Rugby

To write the query, we select all the columns that are required in the output by joining both the tables using INNER JOIN on the common field StudentId.

```
SELECT A.StudentId, A.Art, S.Sport FROM Arts A
INNER JOIN Sports S on A.StudentId = S.StudentId;
```

We get the following output.



StudentId	Art	Sport
1	Painting	Cycling