

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: Annie Sadler Kamanya |
| Course Date: 23/02/25 |
|  |

**Table of contents**

[Day 1: Task 1 3](#_Toc106466110)

[Day 1: Task 2 3](#_Toc1673105856)

[Day 3: Task 1 4](#_Toc1682646377)

[Day 4: Task 1: Written 6](#_Toc135095808)

[Day 4: Task 2: SQL Practical 9](#_Toc1228298484)

[Course Notes 19](#_Toc631049525)

[Additional Information 19](#_Toc1612679949)

# Day 1: Task 1

Please research and complete the below questions relating to key concepts of databases.

|  |  |
| --- | --- |
| What is a primary key? | Primary Key consist of one or more columns in a field that is uniquely identified for each record within a table. This means that no two rows can have the same primary key value, ensuring the uniqueness of every record. Additionally, primary keys cannot contain NULL values.eg of primary key- NHS number, ID number. |
| How does this differ from a secondary key? | * secondary key provides a secondary reference point for objects whose primary keys do not adequately distinguish them for reference purposes. * It is used for identification of rows but not usually unique. * There can be multiple secondary keys per table. * In case primary key is not enough to distinguish an object, a secondary key can be used to render that uniqueness. * Attributes used for Secondary Key are not used for Super key; eg of secondary keys include Street address number, Phone number, Middle name. |
| How are primary and foreign keys related? | Primary keys and foreign keys are two types of constraints that can be used to enforce data integrity in SQL Server tables.  A foreign key is an attribute in one table that contains the value of a primary key in another table. This creates a relational table that links to each order. |
| Provide a real-world example of a one-to-one relationship | **one-to-one relationships** -two items in which one can only belong with the other.  **Real-world eg.**  One passport –one person.  NHS number ID - is unique to one person.  One car- is owned by one person. |
| Provide a real-world example of a one-to-many relationship | **one-to-many relationship** – two items related to one or more records.  **Real-world eg.**  One customer –making many order  One Mother – many children |
| Provide a real-world example of a many-to-many relationship | **Many-to-many relationship** occurs when multiple records in one table can be associated with multiple records in another table.  **Real-world eg.**  Many students in a class-enrol in many courses  An individual belongs to multiple groups- a group may contain multiple people. |

# Day 1: Task 2

Please research and complete the below questions relating to key concepts of databases.

|  |  |
| --- | --- |
| What is the difference between a relational and non-relational database? | Relational and non-relational databases are used by businesses to store, retrieve, and analysis data.  Relational databases are suitable for storing, retrieving, and manipulating well-defined, structured data. Relational database stores data in tables composed of rows and column, IT contained within a table, which is then linked to data contained within other tables using unique identifying keys. Specifically, relationships between tables are formed when a primary key, which uniquely identifies a row in one table, connects with a foreign key identifying a row of data in another table.  Relational databases are designed to store structured data or well-defined data like names, dates, and quantities that can be easily standardized within a table. SQL is the most common programming language used to interface with relational databases within relational database management systems (RDMS).  **Common relational databases**  MySQL  IBM Db2  Microsoft Access  PostgreSQL  Microsoft SQL Server  Oracle  A non-relational database is a type of database that doesn’t store data in tables but instead in whatever format is best for the type of data being stored. In effect, non-relational databases are designed to contain unstructured data, or loosely defined data like email messages, videos, images, and business documents that are not easily standardized. They can also be used to store a mixture of structured and unstructured data.  Non-relational databases are said to be NoSQL, meaning that they do not use Structured Query Language, even though many NoSQL databases do support SQL queries.  **Key-value stores**: In a key-value store, data is assigned a unique identifier, which allows it to be retrieved and sorted. The system consists of "keys," or unique identifiers like a string of numbers, and the "values" with which they are associated, which can be either data itself or simply its location within the database.  **Column-family data stores**: In a column-family data store, data is organized in a "key space" containing multiple families of different columns. The column families, in turn, include rows that each also contain columns containing data. The data can be accessed and identified using keys.  **Graph databases**: Graph databases store data in nodes and structures them based on their relationships to one another, allowing for a much more flexible schema than can be found in tabular designs.  **Document databases:** Document databases store data within documents, which typically contain one object and all its associated metadata.  **Most common non-relational databases include**:  MongoDB  IBM Cloundant  Amazon DynamoDB  Apache Cassandra |
| What type of data would benefit off the non-relational model?  Why? | Non-relational databases can handle a wide range of data types and formats, making them ideal for handling unstructured and semi-structured data, such as social media feeds or sensor data.  As a result, non-relational databases do not follow a rigid schema, instead have more flexible structures to accommodate their data types. Non –relational database uses varying query languages (some do not even have a query language).  Because of their unstructured type, they are scalable horizontally, which allows the use of multiple nodes in a cluster to handle increased workloads. |

# Day 3: Task 1

Please research the below ‘JOIN’ types, explain what they are and provide an example of the types of data it would be used on.

|  |  |
| --- | --- |
| Self-join | **Self JOIN** is a regular join, but the table is joined with itself.  Eg. of self join syntax  Select column\_name(s) From table1 T1, table1 T2 Where condition; |
| Right join | **Right JOINs** are logically the opposite of Left JOINs—they return all rows from the second table and only the rows in the first table that match, eg of how to use a Right Outer Join clause to join two tables:  Select \* From table1 Right Outer Join table2 On table1.id = table2.user\_id |
| Full join | **Full JOINs** combine left and right joins by returning all rows from both tables as long as at least one match exists between them. Example of how to use a Full Outer Join clause to join two tables:  Select 0 \* From table1 Full Outer Join table2 On table1.id = table2.user\_id; |
| Inner join | **Inner JOINs** combine two tables based on a shared key. For example, if you had a table with a column called "user id" and each user ID was unique to a user, you could join that table to another table with a "user id" column to find the information associated with each user; eg  Select \* From table1 Inner Join table2 On table1.id = table2.id; |
| Cross join | **Cross JOINs**- return all records from both tables, whether the other table match or not, this can potentially return very large results. Example  Select column\_name(s), From table1 Cross Join table2; |
| Left join | **Left JOINs-** return all rows from the first table and only those in the second table that match; example of how to use a Left Outer JOIN clause to join two tables:  Select \* From table1 Left Outer Join table2 On table1.id = table2.user\_id; |

# Day 4: Task 1: SQL Practical

In your groups, work together to answer the below questions. It may be of benefit if one of you shares your screen with the group and as a team answer / take screen shots from there.

**Setting up the database:**

1. **Visit this page** [**SQLBolt - Learn SQL - SQL Lesson 1: SELECT queries 101**](https://sqlbolt.com/lesson/select_queries_introduction)

**Write your query using the table given. Copy paste your query in the box**

**Once written query is correct, green tick will appear next to the task.**

**Remove written query to view original table.**

1. **Find the title of each film**

|  |
| --- |
| Select title from movies; |

1. **Find the director of each film**

|  |
| --- |
| Select director from movies; |

1. **Find the title and director of each film**

|  |
| --- |
| Select title, director  from movies; |

1. **Find the title and year of each film**

|  |
| --- |
| Select title, year from movies; |

1. **Find all the information about each film**

|  |
| --- |
| Select \* from movies; |

1. **Find the movie with a row id of 6**

|  |
| --- |
| SELECT \* from movies where rowid = 6; |

1. **Find the movies released in the years between 2000 and 2010**

|  |
| --- |
| Select \* from movies where year between '2000' AND '2010;' |

1. **Find the movies not released in the years between 2000 and 2010**

|  |
| --- |
| Select \* from movies where year Not between 2000 AND 2010; |

1. **Find the first 5 Pixar movies and their release year**

|  |
| --- |
| SELECT \* from movies where year limit 5; |

1. **Find all the Toy Story movies**

|  |
| --- |
| SELECT title FROM movies WHERE title LIKE "toy story%”; |

1. **Find all the movies directed by John Lasseter**

|  |
| --- |
| Select title From movies WHERE director = 'John Lasseter'; |

1. **Find all the movies (and director) not directed by John Lasseter**

|  |
| --- |
| SELECT title FROM movies WHERE Not director = 'John Lasseter'; |

1. **Find all the WALL-\* movies**

|  |
| --- |
| SELECT \* from movies where TITLE like 'Wall-%'; |

1. **List all directors of Pixar movies (alphabetically), without duplicates**

|  |
| --- |
| select distinct director from movies order by director; |

1. **List the last four Pixar movies released (ordered from most recent to least)**

|  |
| --- |
| Select distinct title From movies order by year desc limit 4; |

1. **List the first five Pixar movies sorted alphabetically**

|  |
| --- |
| select title from movies order by title limit 5; |

1. **List the next five Pixar movies sorted alphabetically**

|  |
| --- |
| select title from movies order by title limit 5 offset 5; |

1. **Find the domestic and international sales for each movie**

|  |
| --- |
| SELECT title, domestic\_sales, international\_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id; |

1. **Show the sales numbers for each movie that did better internationally rather than domestically**

|  |
| --- |
| Select title, international\_sales, domestic\_ sales from movies join boxoffice on movies.id = boxoffice.movies.id where international\_sales > domestic\_sales; |

1. **List all the movies by their ratings in descending order**

|  |
| --- |
| Select title, rating from movies join boxoffice on movies.id = boxoffice.movies\_id order by rating desc; |

|  |
| --- |
| **Course Notes** |

It is recommended to take notes from the course, use the space below to do so, or use the revision guide shared with the class:

Normalization in SQL is a process of organizing data within a database (relational database). It involves breaking down a large complex table into smaller and simpler tables whiles maintaining data relationships

Importance

-reduces redundancy (reduce duplicating information).

-improves query performance

-Minimizes update anomalies

**There are 3 xpossible Relationship between entities**

One-to-one relationship eg one passport to one person.

One-to-many relationship eg Mother with many children

Many-to-many relationship eg Many students in a class-enrolls in many courses

To be able to create a relationship, one needs a primary key.

Primary key is a field that uniquely identifies each record in a table.

Foreign key is a column in one table that links to the primary key to another table.

|  |
| --- |
| **Additional Information** |

We have included a range of additional links to further resources and information that you may find useful, these can be found within your revision guide.

**END OF WORKBOOK**

**Please check through your work thoroughly before submitting and update the table of contents if required.**

**Please send your completed work booklet to your trainer.**