Reference card

Database views and normalisation

Database views

SQL queries that return **tables/result sets** can be assigned to a virtual table referred to as a **VIEW**.

Types of views

Below are the types of views and how they may be used.

Look up VIEW

- Primarily used to fetch **static** or **reference data**.
- For example, providing a list of **country names** and their **estimated population** or **GDP**.

```
CREATE VIEW Lookup_view AS
SELECT
        Column1,
        Column2,
        Column3
FROM
        Table_name
```

Join VIEW

- **Combines data** from multiple tables into one dataset.
- Useful for **representing relationships** between tables without requiring complex joins.

```
CREATE VIEW Join_view AS
SELECT
        A.Column1,
        A.Column2,
        B.Column3
FROM
        Table_name_1
AS A
INNER JOIN
        Table_name_2
AS B
ON A.Column_name=B.Column_name
```

Aggregation VIEW

- Used to **display summarised** or **aggregated data**.
- Often involves **GROUP BY** clauses and **aggregation functions** like SUM, AVG, COUNT, etc.

```
CREATE VIEW Aggregationn_view AS
SELECT
        Column1,
        SUM (Column2)
            AS Sum_Column2
        AVG (Column3)
            AS Avg_Column3
FROM
        Table_name_1
```

Database normalisation

Normalisation is a **database design technique** used to organise data in a relational database. It involves **breaking down** a **large**, **complex** table into **smaller**, related tables and **establishing relationships** between them. The process is guided by a set of rules and principles called **normal forms** and helps us **reduce data redundancy** and **address data anomalies**.

Database anomalies

Data anomalies are **unexpected occurrences** in a dataset that can **erode** data **accuracy, reliability**, and **usability**, eventually **undermining data quality** for decision-making, analysis, and reporting.

	Description	Effect
Update anomaly	Occurs in a database with duplicated data where **updating** a piece of information requires modifying multiple rows or **records** in a table.	Can lead to **inconsistencies** or **inaccuracies** in the data.
Insertion anomaly	Occurs when there is difficulty in adding a new record into a table because certain **required** information is not yet **available** for the entity being represented.	Can introduce **incomplete data**.
Deletion anomaly	Occurs when **deleting** a single piece of data results in the unintentional **removal** of **related data** that are still **valid** and **necessary**.	Can cause **loss** of required data.

Normal forms

To **normalise** a database, we must satisfy the **requirements represented by** each of the **normal forms** in a **progressive manner**. That is, in order to have 3NF we must have 2NF, and in order to have 2NF we must have 1NF.

	Criteria	How to convert	Issues addressed
First Normal Form (1NF)	• Data must be **organised in a tabular structure**. • Each cell in the table must **not hold more than one** value **(atomicity)**. • Each table **must have a primary key** for identification.	• If a cell contains **more than one** value, create new rows for each value until each cell contains **one value only**. • **Create** a **primary key** for your table. • Each table **must have a primary key** for identification.	• By enforcing the rule of **atomic values** and organising data in a **structured**, **non-redundant** manner, it helps **prevent update anomalies** and **improve data consistency**.
Second Normal Form (2NF)	• It is already in **1NF**. • It does not contain **partial dependencies**: non-key attributes are fully functionally **dependent** on the **entire primary key**.	• **Identify primary key**: Find the unique identifier for each row, which can be a single attribute or a combination. • **Table splitting** (if needed): If partial dependencies exist, consider dividing the table into related tables, each with its own primary key. • **Create relationships**: When tables are split, establish links using foreign keys in child tables referencing parent table primary keys for data consistency.	• By eliminating partial dependencies it **reduces update anomalies, improves insertion** and **deletion operations**, and promotes structured data organisation.
Third Normal Form (3NF)	• It is already in **2NF**. • It does not contain **transitive dependencies**: non-key attributes are not transitively dependent on the primary key through other non-key attributes.	• Eliminate transitive dependencies by creating **separate tables** for the attributes causing the **transitive dependency**. • Link the tables using **foreign keys**.	• By enforcing **direct dependencies**, it addresses data anomalies associated with transitive dependencies on the primary key, such as **reducing update** and **deletion anomalies**.