Full Key Dependencies: When an attribute is fully dependent on the attribute and not on any of its proper subset.

Partial Key Dependencies: When one primary key determines some other attributes.

Transitive dependency: When non-key attribute determines some other attributes.

Reason for normalization:

1. Redundancy: Inefficient storage
2. Anomalies: data inconsistency, difficulties in maintenance

Nomalizations

1. 1NF : All attributes depend on the key

* Every relational table is, by definition, 1NF. States each value in each column of a table must be a single value from the domain of the column

1. 2NF: All attributes depend on the whole key

* 1NF with no possibility of partial dependency

1. 3NF: All attributes depend on nothing but the key

* Does no contain transitive functional dependencies

1. BCNF: No functional dependencies other than full key functional dependencies

Table created to find composite keys (Candidate Keys)

A picture containing text, screen

Description automatically generated

**SQL Queries**

**WHERE**

determines which rows should be retrieved. Logical operatorsr can be used like = > < <= >= != <>

Multiple comparison expressions can be used for WHERE statement

**GROUP** **BY**

Enables summarization across the groups of related data within tables. Columns selected from GROUP BY clause should be listed on the SELECT clause as well

**HAVING**

Enables summarization across the group of related data within tables. Determines which groups will be displayed in the result of query. Must be used with GROUP BY clause

Ex.   
SELECT RegionID, COUNT(StoreID)   
FROM Store   
GROUP BY RegionID   
HAVING COUNT(StoreID) >= 4;

**ORDER** **BY**

To sort the results of the query by one or more columns, ORDER BY clause within the used SELECT query. ORDER BY sorts the data in ascending order. For descending order, use keyword DESC.

**\***

Wild card that stands for “All Columns”

**COUNT, MIN, MAX, SUM, AVG**

Numeral expression for calculation used in query. Typically only works in SELECT clause.

**AS**

Alias used for renaming columns displayed. Used in SELECT clause  
ex. COUNT(CustomerID) AS numberOfCustomers

**TABLES**

Create tables with “CREATE TABLE” key word and declaring columns with keys, values, names, and data type. Key value is optional.

Ex.

CREATE TABLE Product(

ProductID char(3) PRIMARY KEY,

ProductName varchar(35) NOT NULL,

ProductPrice int NOT NULL,

VendorID char(2) NOT NULL,

CategoryID char(2) NOT NULL,

FOREIGN KEY(VendorID) REFERENCES Vendor(vendorid),

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

**INSERT**

Use INSERT Clause to insert information into the given table. Format goes as follows: INSERT INTO ‘tableName’(column) VALUES (‘content’). Order of the column must match the order written in VALUES

Ex. INSERT INTO vendor(vendorid,vendorname) VALUES ('PG','Pacifica Gear');