Super Keys:

- A **super key** is a set of one or more attributes, which can uniquely identify a row in a table.
- A super key may have additional attributes that are not needed for unique identification.
- **E.g. 1.** Suppose that we have a relation called Students with the attributes id, first_name, last_name and average and {id} is a super key.
  Since {id} is a super key, then the following are also super keys:
  - {id, first_name}
  - {id, last_name}
  - {id, average}
  - Etc

  This is because since id can uniquely identify a row in Students, anything else we add to the set can also uniquely identify a row in Students.
- Another way to define super keys is through closure. The closure of a super key should give back the entire relation.
- **E.g. 2.** Given R(A, B, C) and A → BC.
  The closure of A, A^+ = {A, B, C}.
  Since the closure of A gives back the entire relation R, it is a super key.
- **E.g. 3.** Given R(A, B, C, D) and
  ABC → D
  AB → CD
  A → BCD

  What are the super key(s) if any exist?

  Soln:
  $(ABC)^+$ = {A, B, C, D}
  $(AB)^+$ = {A, B, C, D}
  A^+ = {A, B, C, D}
  In this example, {A}, {A, B} and {A, B, C} are all super keys. There are more.

Candidate Keys:

- A **candidate key** is a minimal super key.
  I.e. It is the minimal set of attributes needed to uniquely identify a row in a table.
  In example 1, only {id} is a candidate key.
  In example 3, only {A} is a candidate key.
- Properties of candidate keys:
  - It must contain unique values.
  - It may have multiple attributes.
  - It must not contain null values.
  - It should contain the minimum fields to ensure uniqueness.
  - It should uniquely identify each record in a table.
- A table can have multiple candidate keys.

- **E.g. 4.** Given R(A, B, C, D) and
  B → ACD
  ACD → B

  List out all the candidate keys, if there are any.

  Soln:
  B^+ = {A, B, C, D}
  $(ACD)^+$ = {A, B, C, D}.
  Hence, both {B} and {A, C, D} are both super keys.
  However, because they are both minimal, they are also candidate keys.
  Note that for ACD, you cannot break it down and still get all the attributes in R.
  A^+, C^+, D^+, $(AC)^+$, $(AD)^+$, $(CD)^+$ do not give you all the relations in R.
  Hence, ACD is minimal.
  So in this case, we have 2 candidate keys for the relation R.
- **E.g. 5.** Given R(A, B, C, D) and
  AB → C
  C → BD
  D → A

  List all the candidate keys, if there are any.

  Soln:
  $(AB)^+$ = {A, B, C, D}
  C^+ = {A, B, C, D}
  D^+ = {A, D}

  In this example, both {A, B} and {C} are candidate keys.
- **E.g. 6.** Given R(A, B, C, D) and
  A → B
  B → C
  C → A

  List all the candidate keys, if there are any.

  Soln:
  First, notice that neither A, B nor C can get you column D. Hence, we know that our candidate key must contain D.

  A^+ = {A, B, C}
  B^+ = {A, B, C}
  C^+ = {A, B, C}

  Hence, the candidate keys are {A, D}, {B, D}, and {C, D}.

- **E.g. 7.** Given R(A, B, C, D) and
  AB → CD
  D → A
  List all the candidate keys, if there are any.

  Soln:
  $(AB)^+$ = {A, B, C, D}
  D^+ = {A, D} ← Notice that the closure of D has all the relations except for B and C. We know that AB gets us B and C and we already have A.
  $(BD)^+$ = {A, B, C, D}

  Hence, {A, B} and {B, D} are candidate keys.
- **E.g. 8.** Given R(A, B, C, D, E, F) and
  AB → C
  C → D
  B → AE

  List all the candidate keys, if there are any.

  Soln:
  $(AB)^+$ = {A, B, C, D, E}
  B^+ = {A, B, C, D, E} ← Only missing column F.
  C^+ = {C, D}

  Hence, {B, F} is the only candidate key.
- **E.g. 9.** Given R(A, B, C, D) and
  AB → CD
  C → A
  D → B

  List all the candidate keys, if there are any.

  Soln:
  $(AB)^+$ = {A, B, C, D}
  C^+ ={A, C} ← Missing B and D. We know that AB gets us CD, so {B, C} is a candidate key as we already have A.
  D^+ = {B, D} ← Missing A and C. We know that AB gets us CD, so {A, D} is a candidate key as we already have B.

  Hence, {A, B}, {B, C}, {C, D} and {A, D} are the candidate keys.

Primary Keys:

- A **primary key** is a chosen candidate key.
  I.e. There could be multiple candidate keys. From the options, we choose one to use. The one that we chose to use is the primary key.
- Rules for defining primary keys:
  - Two rows can't have the same primary key value.
  - The primary key field cannot be null.
  - The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.
- **Prime attributes** are the attributes of the primary key.
- **Non-prime attributes** are the attributes of a table not in the primary key.

Normalization:

- **Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like insertion, update and deletion anomalies.
- Normalization divides larger tables into smaller tables and links them using relationships.
- The purpose of normalization is to eliminate repetitive data and ensure data is stored logically.
- **E.g. 10.** Consider the table below:

  Student

SID	Name	Program	Department Head	Department Head's Phone Number
1	A	CSC	X	100-100-1000
1	A	MAT	Y	100-100-1001
2	B	CSC	X	100-100-1000
3	C	MAT	Y	100-100-1001
4	D	STA	Z	100-100-1002

  Here are some problems with this design:

  1. Suppose we enroll a new student who's not in any program. Then, the program, department head and department head's phone number will be blank. This is an example of **insertion anomaly**.
  2. Suppose that a department head gets changed. Then, we would have to change that information for multiple students, and if by mistake we miss any record, it will lead to data inconsistency. This is an example of **updation anomaly**.
  3. We see that the department head and department head's phone number information are repeated for the students who are in that program. This is an example of **data redundancy**.
  4. Suppose that student D graduated and all rows pertaining to student D gets deleted. If student D is the only student in the stats program, then we lose important information, such as student D's program, the program's department chair and the department chair's phone number, when we delete all rows pertaining to student D. This is an example of **deletion anomaly**.

- Anomalies are caused when there is too much redundancy in the database's information.
- **Update anomaly** happens when there are multiple entries of the same data in the db and when we update that data, one or more entries do not get updated. Then, we will have data inconsistency.
- **Insertion anomaly** happens when inserting vital data into the database is not possible because other data is not already there.
- **Deletion anomaly** happens when the deletion of unwanted information causes desired information to be deleted as well.
- There are a few normalization rules we can use:
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce and Codd Normal Form)

- **1NF (First Normal Form):**
- For a table to be in the First Normal Form, it must follow the following rules:
    1. Each table cell should contain a single value.
    2. Each record needs to be unique.
    3. Values stored in a column should be of the same domain
    4. All the columns in a table should have unique names.
- **2NF (Second Normal Form):**
- For a table to be in the Second Normal Form, it must follow the following rules:
    1. It is already in First Normal Form.
    2. It should not have **partial dependency**. **Partial dependency** occurs when an attribute in a table depends on only a part of the primary key and not on the whole key.
       E.g. 11. Consider R(A, B, C, D) and
       AB → D
       B → C

       We see that the primary key is {A, B}. However, R is not in 2NF because the attribute C only depends on B and not A & B. This is an example of partial dependency.

       To change R to 2NF, we have to decompose it so that the partial dependencies are its own tables.

       For this example, we decompose R(A, B, C, D) into
       R1(A, B, D) and
       R2(B, C)
       Note: When you decompose R into smaller relations, you always want a relation with the primary keys. In this case, we have R1, so we don't need an additional table.
- **E.g. 12.** Consider R(A, B, C) and
  AB → C
  B → C

  We see that the primary key is {A, B}. We see that B → C is a partial dependency.

  To change R to 2NF, we have to decompose it so that the partial dependencies are its own tables.

  For this example, we decompose R(A, B, C) into
  R1(A, B) and
  R2(B, C)

  We don't have C in R1 because we already have C in R2.
- **E.g. 13.** Consider R(A, B, C, D, E) and
  AB → C
  D → E

  We see that the primary key is {A, B, D}. We see that AB → C and D → E are partial dependencies.

To change R to 2NF, we have to decompose it so that the partial dependencies are its own tables.

For this example, we decompose R(A, B, C, D, E) into
R1(A, B, C)
R2(D, E)
R3(A, B, D)

- **E.g. 14.** Consider R(A, B, C, D, E) and
A → B
B → E
C → D

We see that the primary key is {A, C}. We see that A → B and C → D are partial dependencies.

To change R to 2NF, we have to decompose it so that the partial dependencies are its own tables.

For this example, we decompose R(A, B, C, D, E) into
R1(A, B, E)
R2(C, D)
R3(A, C)

Note: B → E is not a partial dependency as B is not part of the primary key. For 2NF, we simply find the relation that contains B, which is R1, and add E to it.

- **3NF (Third Normal Form):**
- A table is in third normal form if:
    1. It is in 2nd normal form.
    2. It must not have **transitive dependencies**.

Recall: A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.
I.e. If A → B and B → C, then A → C is a transitive dependency.
Another way to think about transitive dependency is that it occurs when a non-prime attribute depends on other non-prime attributes.

- The normalization of 2NF relations to 3NF relations involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

Recall: The left side of a functional dependency is called the determinant.

- **E.g. 15.** Consider R(A, B, C) and
A → B
B → C

We see that A^+ = {A, B, C}, so {A} is the primary key and A is a prime attribute.
Furthermore, we see that we have a transitive dependency B → C.
What we do is we split R into 2 relations:
R1(A, B)
R2(B, C)

- Let P be a prime attribute and NP be a non-prime attribute and suppose that {P} is not a primary key. Then, we have
    1. **Partial Dependency** if we have P → NP.
    2. **Transitive Dependency** if we have NP → NP.

  If we have P/NP → P, we know for sure that it'll be in 3NF.
- **E.g. 16.** Consider R(A, B, C, D, E) and
  A → B
  B → E
  C → D

  We see that $(AC)^+$ = {A, B, C, D, E}, so {A, C} is a primary key.
  We see that we have
    1. A → B (Partial dependency)
    2. C → D (Partial dependency)
    3. B → E (Transitive dependency)

  To turn R into 3NF, we will break it down into the following relations:
  R1(A, B, E) ← Since B → E, we put E here. However, B → E is a transitive dependency, so we have to split up R1. We will split R1 up into R11 and R12.
  R11(A, B)
  R12(B, E)
  R2(C, D)

  R3(A, C) ← **Note:** When you decompose R into smaller relations, you always want a relation with the primary keys. In this case, we need to create a new relation to get a relation with the primary keys.

  The final decomposition of R is:
  R11(A, B)
  R12(B, E)
  R2(C, D)
  R3(A, C)
- **E.g. 17.** Consider R(A, B, C, D, E, F, G, H, I, J) and
  AB → C
  A → DE
  B → F
  F → GH
  D → IJ

  A primary key is {A, B} as the closure of (AB) gets back all attributes in R.
  We see that
    1. A → DE is a partial dependency (pd).
    2. B → F is a pd.
    3. F → GH is a transitive dependency (td).
    4. D → IJ is a td.

  We want to decompose R so that it is in 3NF.
  We start with R1(A, D, E, I, J). We know that D → IJ, so we put I and J here. However, D → IJ is a td, so we have to split up R1 into R11 and R12.
  R11(A, D, E)
  R12(D, I, J)

Next, we have R2(B, F, G, H). We know that F → GH, so we put G and H here.
However, F → GH is a td, so we split up R2 into R21 and R22.
R21(B, F)
R22(F, G, H)

R3(A, B, C)

The final decomposition of R is:
R11(A, D, E)
R12(D, I, J)
R21(B, F)
R22(F, G, H)
R3(A, B, C)

- **E.g. 18.** Consider R(A, B, C, D, E) and
AB → C
B → D
D → E

A primary key is {A, B} as the closure of (AB) gets back all attributes in R.
We see that B → D is a pd and that D → E is a td.
We need to decompose R.
We start with R1(B, D, E). We know that D → E, so we put E here. However, D → E is a td, so we have to split R1 into R11 and R12.
R11(B, D)
R12(D, E)

Next, we have R2(A, B, C).

The final decomposition of R is:
R11(B, D)
R12(D, E)
R2(A, B, C)

- **BCNF (Boyce and Codd Normal Form):**
- For a table to be in BCNF, following conditions must be satisfied:
    1. It must be in 3NF.
    2. For each functional dependency (X → Y), X should be a super key.
- **E.g. 19.** Consider R(A, B, C) with the fds
AB → C
C → B

We see that {A, B} and {A, C} are candidate keys.
Hence, A, B and C are all prime attributes.
AB → C is neither a pd nor td.
C → B is neither pd or td because both the LHS and the RHS have prime attributes.
Hence, we see R is in 3NF.
However, R is not in BCNF because in C → B, C is not a super key.
To fix this, I'll decompose R into
R1(C, B)
R2(A, C) ← We chose (A, C) over (A, B) to prevent loss of data when joining R1 and R2.

- **E.g. 20.** Given R(A, B, C, D, E, F, G, H) and
  AB → C
  A → DE
  B → F
  F → GH

  What form is it?

  Soln:
  We see that a candidate key is {A, B}.
  We see that A → DE is a pd. Hence, R is in 1NF only.
- **E.g. 21.** Given R(A, B, C, D, E) and
  CE → D
  D → B
  C → A

  What form is it?

  Soln:
  We see that a candidate key is {C, E}.
  We see that C → A is a pd. Hence, R is in 1NF only.