1.

a.

PERSON Primary Key: NickName. There are no alternate keys since duplicates of FullName and BirthDate are possible.

POST Primary Key: PostID. Another candidate key is (PersonNick, Datetime). The pairing of these two is always unique since it is impossible for the same person to post something multiple times at the same time. Individually, the attributes can have duplicates, meaning this is an appropriate candidate key. However, because uniqueness can already be determined by the single PostID attribute, the PostID attribute is the better primary key.

FRIENDS Primary Key: (PersonNick1, PersonNick2). These attributes must be paired since it is possible for a single person to have many friends. However, each pairing of these attributes will be unique. Because there are no other attributes, there are no other candidate keys.

LIKE Primary Key: (PostID, PersonNick). These attributes must be paired since it is possible for multiple likes to exist on a single post and since it is possible for the same person to like posts more than once. However, a person can only like a single post once, meaning that each PostID and PersonNick pairing will be unique. Alternatively, one could use the pairing (PersonNick, Datetime) since it would be impossible for the same person to like something multiple times at the exact same time. The more appropriate primary key would be (PostID, PersonNick) since it not only uniquely identifies specific likes, but also helps reference liked posts.

LIKETYPE Primary Key: LikeType. This numeric attribute is used to uniquely identify specific like types. Another candidate would be LikeName, since it is presumably has a one-to-one association with its LikeType. Both keys work well as primary keys. However, LikeType might be slightly better since it is a numeric attribute.

COMMENT Primary Key: CommentID. Another possible candidate key would be (PostID, PersonNick, Datetime). Not one of these attributes alone can uniquely identify a comment since many comments can exist for a single post, a single person can comment many times, and because many comments can be made at the same time. However, it is impossible for a single person to comment on a single post multiple times at the same time. Despite the inherent uniqueness in this combination, the single attribute CommentID is the better pick for uniquely identifying comments.

PRIVATE_MESSAGE Primary Key: MessageID. Other candidate keys are (PersonNick1, Datetime) and (PersonNick1, PersonNick2, Datetime). Whether a message is sent by one person to anyone or sent by one person to another specific person, these messages will never be sent by the same person multiple times at the exact same time. Despite this uniqueness, MessageID already sufficiently identifies messages without depending on other attributes.

b.

The primary keys I selected would not change. However, the other available candidate keys would change. Any key that relies on Datetime to help uniquely identify that relation would no longer be a viable candidate key

c.

PRIVATE_MESSAGE foreign key PersonNick1 references PERSON primary key NickName.

PRIVATE_MESSAGE foreign key PersonNick2 references PERSON primary key NickName.

COMMENT foreign key PostID references POST primary key PostID.

COMMENT foreign key PersonNick references PERSON primary key NickName.

LIKE foreign key Type references LIKETYPE primary key LikeType.

LIKE foreign key PersonNick references PERSON primary key NickName.

LIKE foreign key PostID references POST primary key PostID.

FRIENDS foreign key PersonNick1 references PERSON primary key NickName.

FRIENDS foreign key PersonNick2 references PERSON primary key NickName.

POST foreign key PersonNick references PERSON primary key NickName.

POST foreign key WallPersonNick references PERSON primary key NickName.

2. Final Query Answers are Bold

a.

Find all the PostID from posts created by “Mike_34” and liked with “wow”

Let R1 = POST, R2 = LIKE, R3 = LIKETYPE

PostID.1 is PostID from POST while PostID.2 is PostID from LIKE.

PersonNick.1 is PersonNick from POST while PersonNick.2 is PersonNick from LIKE.

RESULT = Project <PostID.1> (R1 Theta Join <PostID.1 = PostID.2 and PersonNick.1 = “Mike_34”> (R2 Theta Join <Type = LikeType and LikeName = “wow”> R3)))

b.

Find the MessageID and the full name of the message’s creator for all the messages that “Mike_34” has unread and that are older than the 23rd December 2018.

Let R1 = PERSON, R2 = PRIVATE_MESSAGE

Let Read value of read be “read” and Read value of unread be “unread”.

RESULT = Project <MessageID, FullName> (R1 Theta Join <NickName = PersonNick1 and PersonNick2 = “Mike_34 and Read = “unread”> R2)

c.

Find the FullName of all the friends of persons called “John Norton”

Let R1 = PERSON, R2 = FRIENDS

Let NickName1 = Project <PersonNick1> (R1 Theta Join <(NickName = PersonNick1 or NickName = PersonNick2) and FullName = “John Norton”> R2)

Let NickName2 = Project <PersonNick2> (R1 Theta Join <(NickName = PersonNick1 or NickName = PersonNick2) and FullName = “John Norton”> R2)

Let FriendPair1 = Project <FullName> (R1 Theta Join <PersonNick1 = NickName and FullName != John Norton> NickName1)

Let FriendPair2 = Project <FullName> (R1 Theta Join <PersonNick2 = NickName and FullName != John Norton> NickName2)

RESULT = Project <FullName> (FriendPair1 Natural Join FriendPair2)

d.

Find the postID, Content and the FullName of the post creator, for each post commented by “Jenny_27” that is in “Mike_34” wall.

To differentiate attributes after joining, PostID.1 is PostID from POST while PostID.2 is PostID from COMMENT.

Additionally, PersonNick.1 is PersonNick from POST while PersonNick.2 is PersonNick from COMMENT.

Let PostCollection = Project <PostID.1, Content, PersonNick.1> (POST Theta Join <PostID.1 = PostID.2 and PersonNick.2 = “Jenny_27” and WallPersonNick = “Mike_34”> COMMENT)

RESULT = Project <PostID.1, Content, FullName> (PERSON Theta Join <NickName = PersonNick.1> PostCollection)

e.

Retrieve the names of all the Persons who have at least a (one) post without likes and without comments

Note: “names” does not specify FullName or NickName. This query assumes the retrieval of NickNames.

PostID.1 is PostID from POST. PostID.2 is PostID from UncommentedUnlikedPosts.

Let UncommentedUnlikedPosts = Project <PostID> (Project <PostID> Post) – (Project <PostID> (LIKE Natural Join COMMENT))

RESULT = Project <PersonNick> (POST Theta Join <PostID.1 = PostID.2> UncommentedUnlikedPosts)

f.

Find all persons that liked any post created by “Mike_34” but none of the posts created by “Margaret Lind”

Since I am tasked with finding all persons, the result will be a full relation, not just an attribute.

PostID.1 is PostID from MargaretPosts. PostID.2 is PostID from LIKE. PostID.3 is PostID from MikePosts.

Let MargaretPosts = Project <PostID> (PERSON Theta Join <NickName = PersonNick and FullName = “Margaret Lind”> POST)

Let NamesLikingMargaret = Project <PersonNick> (MargaretPosts Theta Join <PostID.1 = PostID.2> LIKE)

Let MikePosts = Project <PostID> (PERSON Theta Join <NickName = PersonNick and NickName = “Mike_34”> POST)

Let NamesLikingMike = Project <PersonNick> (MikePosts ThetaJoin <PostID.3 = PostID.2> LIKE)

Let NamesLikingMikeNotMargaret = NamesLikingMike – NamesLikingMargaret

RESULT = PERSON Theta Join <NickName = PersonNick> NamesLikingMikeNotMargaret

g.

Find the nicknames that have all the different types of likes on their posts.

PostID.1 is PostID from LIKE. PostID.2 is PostID from AllLikePosts.

Let AllLikePosts = LIKE(PostID.2, Type) Division LIKETYPE(LikeType)

RESULT = Project <PersonNick> (POST Theta Join <PostID.1 = PostID.2> AllLikePosts)