|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Apartments |  | Buildings |  | Tenants |  |
| AptID | Int | BuildingID | Int | TenantsID | Int |
| UnitNumber | Varchar | ComplexID | Int | TenantName | Varchar |
| BuildingID | Int | BuildingName | Varchar |  |  |
|  |  | Address | Varchar |  |  |
| Complexes |  | AptTenants |  | Requests |  |
| ComplexID | Int | TenantID | Int | RequestID | Int |
| ComplexName | Varchar | AptID | Int | Status | Varchar |
|  |  |  |  | AptID | Int |
|  |  |  |  | Description | varchar |

1. Write a SQL query to get a list of tenants who are renting more than 1 apt.

SELECT TenantNmae

FROM Tenants

INNER JOIN

( SELECT TenantID

FROM AptTenants

GROUP BY TenantID

HAVING count(\*) > 1) C

ON Tenants.TenantID = C.TenantID

1. Write a SQL query to get a list of all buildings and the open number of requests (req with “Open” status)

SELECT BuildingName, ISNULL(Count, 0) AS ‘Count’

FROM Buildings

LEFT JOIN

(SELECT Apartments.BuildingID, count(\*) as ‘Count’

FROM Requests INNTER JOIN Apartments

ON Request.AptID = Apartments.AptID

WHERE Requests.Status = ‘Open’

GROUP BY Apartments.BuildingID) ReqCounts

ON ReqCounts.BuildingID = Buildings.BuildingID

1. Building #11 is undergoing a major renovation. Implement a query to close all requests from apartments from this building

UPDATE Requests

SET Status = ‘Closed’

WHERE AptID IN

(SELECT AptID

FROM Apartments

WHERE BuildingID = 11)

1. What are the different types of join.

JOING is used to combine the results of 2 tables. To perform a JOIN, each of the tables must have at least one field that will be used to find the matching records from the other table. The join type defines which records will go into the result set.

1. What is denormalizatoin. Explain pros and cons.

Denormalizatoin is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. By contrast, in traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in the database. Having normalized is great if we want to change data, because we only need to do it in one place. The drawback is if the tales are large, we may spend a long time doing joins on the tables.

Denormalization strikes a compromise. Under denormalization, we decide that we’re okay with some redundancy and some extra effort to update the database in order to get the efficiency advantage of fewer joins.

Cons of denormalizaton:

* Updates and inserts are more expensive.
* Can make update and insert code harder to write
* Data bay be inconsistent. Which piece is correct?
* Data redundancy means more storage space.

Pros of denormalization:

* Retrieving data is faster since we need fewer joins.
* Queries to retrieve can be simpler and therefore less likely to have bugs since we need fewer tales.