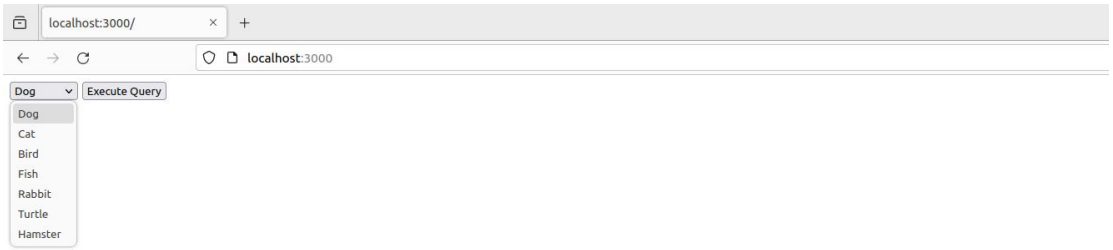


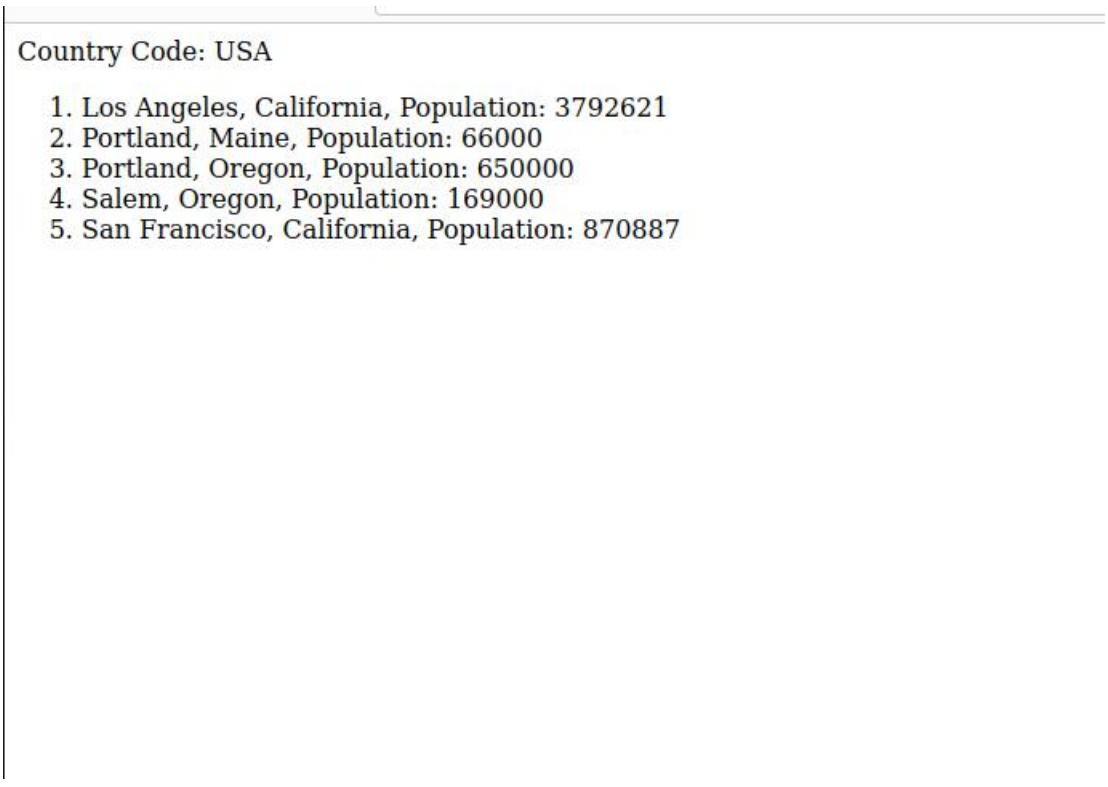
Pet Type: fish

1. 7, Goldie, Goldfish
2. 8, Nemo, Clownfish

Pet Type: hamster

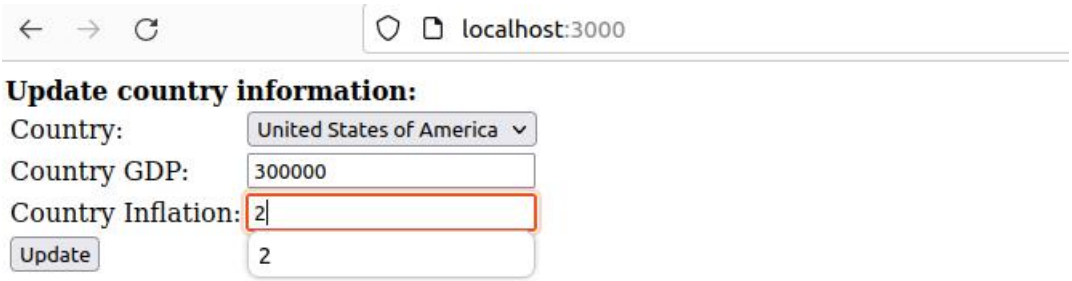
1. 13, Hammy, Syrian Hamster
2. 14, Fuzzy, Dwarf Hamster


United States of America
Execute Query
United States of America
Canada
Brazil
Argentina
Mexico
FictionLand
Islandia
RichLand
PoorVille


Country Code: USA
1. Los Angeles, California, Population: 3792621
2. Portland, Maine, Population: 66000
3. Portland, Oregon, Population: 650000
4. Salem, Oregon, Population: 169000
5. San Francisco, California, Population: 870887

Country Code: BRA

1. Sao Paulo City, Sao Paulo, Population: 12038175



Updated Country Information

- ABC - RichLand, GDP: 300000.00, Inflation: 2.00
- ARG - Argentina, GDP: 113333.00, Inflation: 2.00
- BRA - Brazil, GDP: 1300000.00, Inflation: 3.50
- CAN - Canada, GDP: 1700000.00, Inflation: 2.00
- CBA - PoorVille, GDP: 900000.00, Inflation: 4.50
- FCT - FictionLand, GDP: 3000000.00, Inflation: 3.00
- ISL - Islandia, GDP: 520000.00, Inflation: 1.00
- MEX - Mexico, GDP: 1200000.00, Inflation: 4.00
- USA - United States of America, GDP: 300000.00, Inflation: 2.00

1.
CustomerID (Primary Key)
OrderID (Primary Key)
CustomerName
CustomerEmail
OrderDate
ProductID
ProductName
ProductPrice

(a) Functional Dependencies:

CustomerID -> CustomerName, CustomerEmail
OrderID -> OrderDate, CustomerID
ProductID -> ProductName, ProductPrice

In this scenario, we have three functional dependencies:

CustomerID uniquely determines CustomerName and CustomerEmail.

OrderID uniquely determines OrderDate and CustomerID.
ProductID uniquely determines ProductName and ProductPrice.

(b)

Customer ID	Customer Name	Customer Email	Order ID	Order Date	Product ID	Product Name	Product Price
101	A	A@example.com	001	2023-01-15	P001	CD	1100
101	A	A@example.com	002	2023-01-16	P002	DC	2000
102	B	B@example.com	003	2023-02-10	P003	AC	3300
102	B	B@example.com	004	2023-03-15	P004	AD	4000

In this instance, you can see that customer information (CustomerName and CustomerEmail) is repeated for the same CustomerID and order information (OrderDate) is repeated for the same OrderID. This redundancy can lead to data anomalies.

(c)
Update Anomaly:
If you need to update a customer's email address from A@example.com to alice.new@example.com, you would need to update multiple rows with the same CustomerID, which could lead to inconsistencies or errors if not done correctly.

Insertion Anomaly:
Suppose you want to add a new order for a customer who hasn't placed any orders before. You would need to duplicate the customer's information (CustomerName and CustomerEmail) in the new row, which can lead to unnecessary data redundancy and potential data entry errors.

Deletion Anomaly:
If you delete a row associated with a particular customer (e.g., CustomerID 101), you would also inadvertently remove information about their orders (OrderID 001 and 002), leading to a loss of data and potentially impacting order history records.

2.
(a) . Show that if $X \rightarrow Y$ and $YW \rightarrow Z$, then $XW \rightarrow Z$.
$X \rightarrow Y$
$XW \rightarrow YW$ (Augmentation)
$YW \rightarrow Z$
$XW \rightarrow Z$(Transitivity)
(b) . Show that if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.
Since $Z \subseteq YZ$, we can also infer $YZ \rightarrow Z$.(Reflexivity:)

X→ YZ and YZ→ Z so X→Z(Transitivity)

(c). Show that if X → Y and X → Z, then X → Y Z.
Augment X → Y with Z on both sides to get XZ → Y Z.(Augmentation)
Since X ⊆ XZ, we have X → XZ.(Reflexivity)
From X → XZ and XZ → Y Z, we infer X → Y Z.(Transitivity)

3.
FDs can be used to determine a relation's candidate keys. In particular, a candidate key is a minimal set of attributes that all other attributes functionally depend on. We can test if a set of attributes is a superkey by repeatedly applying the rules in question 1. For example, given the relation R(a, b, c) and the FDs a → b and b → c, we can apply transitivity to determine that a → c, reflexivity to determine that a → a, and a → b is given. Thus a is a candidate key since it is minimal and since a → abc. Using a similar approach, find the candidate keys for the following.
(a) . A relation R(a, b, c) and FDs {a → c}
Because a→c but lose b
So we need add b in key to determine c
ab→c
So ab is candidate key
(b) . A relation R(a, b, c, d) and FDs {b → c, d → a}
b→c and d→a so bd→ac
So bd is candidate key

(c) . A relation R(a, b, c, d) and FDs {a → c, c → d}
a→c and c→d shows a→cd but lose b
So we need add b in key this means ab→cd
ab is candidate key
(d) . A relation R(a, b, c, d, e) and FDs {c → b, bd → e, a → d, e → a}
e→a and bd→e this means bd→ea
But we lose c in this relation so we need add c in key
bdc→ea
So bdc is candidate key.

4.

1. Albums
Relational Schema: Albums(AlbumTitle, YearRecorded, GroupName, RecordLabel)
Primary Key: (AlbumTitle, GroupName)
Functional Dependencies:
(AlbumTitle, GroupName) →YearRecorded
(AlbumTitle, GroupName) → RecordLabel
Normal Form: Since the primary key functionally determines all other attributes and there are no transitive dependencies or partial dependencies, this table is in 3rd Normal Form (3NF).

2. Music Groups
Relational Schema: MusicGroups(GroupName, YearFormed)
Primary Key: GroupName
Functional Dependencies:
GroupName →YearFormed
Normal Form: This relation is in 3NF as every non-key attribute is fully functionally dependent on the primary key.
3. Group Members (Artists)
Relational Schema: Artists(ArtistName, BirthYear, GroupName, StartYear, EndYear)
Primary Key: (ArtistName, GroupName, StartYear)
Functional Dependencies:
(ArtistName, GroupName, StartYear) → BirthYear
(ArtistName, GroupName, StartYear) → EndYear
Normal Form: This table is in 3NF. All non-key attributes are fully functionally dependent on the primary key.
4. Music Genres
Relational Schema: Genres(GenreLabel, Description)
Primary Key: GenreLabel
Functional Dependencies:
GenreLabel → Description
Normal Form: The Genres table is in 3NF as the non-key attribute (Description) is fully functionally dependent on the primary key.
5. Tracks
Relational Schema: Tracks(TrackID, YearRecorded, TrackName, AlbumTitle, GroupName)
Primary Key: TrackID
Functional Dependencies:
TrackID →YearRecorded
TrackID → TrackName
TrackID → AlbumTitle
TrackID → GroupName
Normal Form: The Tracks table is in 3NF. Each attribute is fully functionally dependent on the primary key.
6. Songs
Relational Schema: Songs(SongTitle, YearWritten)
Primary Key: SongTitle
Functional Dependencies:
SongTitle → YearWritten
Normal Form: The Songs table is in 3NF since YearWritten is fully functionally dependent on the primary key.
7.Influence of Music Groups on Each Other
Relational Schema: Influences(InfluencingGroupName, InfluencedGroupName)
Primary Key: (InfluencingGroupName, InfluencedGroupName)
Functional Dependencies: None
Corresponding Normal Form: Since there are no non-trivial functional dependencies involving

non-key attributes, this table is in the Third Normal Form (3NF).