**Question 1:**

1 Attribute Closures

{ArrivalTime}+ = ArrivalTime, DepartureTime, {DepartureTime}+ = DepartureTime, {Location}+ = Location, {Bus}+ = Bus

2 Attribute Closures

{ArrivalTime, DepartureTime}+ = {ArrivalTime, DepartureTime},

{ArrivalTime, Location}+ = {ArrivalTime, DepartureTime, Location, Bus},

{ArrivalTime, Bus}+ = {ArrivalTime, DepartureTime, Location, Bus},

{DepartureTime, Location}+ = {DepartureTime, Location, Bus},

{DepartureTime, Bus}+ = {DepartureTime, Bus},

{Location, Bus} + = {Location, Bus}.

3. Three Attributes Closures

{ArrivalTime, DepartureTime, Location}+ = {ArrivalTime, DepartureTime, Location, Bus}.

{ArrivalTime, DepartureTime, Bus}+ = {ArrivalTime, DepartureTime, Location, Bus}.

{ArrivalTime, Location, Bus}+ = {ArrivalTime, DepartureTime, Location, Bus}.

{DepartureTime, Location, Bus}+: {ArrivalTime, DepartureTime, Location, Bus}.

4. Full Set Closure

{ArrivalTime, DepartureTime, Location, Bus}+ = {ArrivalTime, DepartureTime, Location, Bus}.

**Question 2:**

The minimal set of attributes that can uniquely identify all other attributes in the relation Route is {ArrivalTime,Bus} and {ArrivalTime, Location}. So therefore {ArrivalTime,Bus} is identified as the key for the relation Route.

**Question 3:**

The non-trivial functional dependencies for the relation Route are directly given by the initial functional dependencies:

{DepartureTime, Location} → {Bus}

{ArrivalTime} → {DepartureTime}

{ArrivalTime -> Location} -> {Bus}

{Bus, ArrivalTime} → {Location}

{ArrivalTime, DepartureTime, Location} -> {Bus}  
{ArrivalTime, Location, Bus} ->{DepartureTime}  
{ArrivalTime, DepartureTime, Bus} -> {Location}

**Question 4:**

Given the functional dependency: {Bus, DepartureTime}→{Driver}

To compute {Bus, DepartureTime}+{Bus, DepartureTime}+, we use transitivity and reflexivity:

1. **Given**: {Bus, DepartureTime}{Bus, DepartureTime}
2. **From FD 1**: {Bus, DepartureTime}→{Driver} gives us {Driver}

So,{Bus, DepartureTime}+={Bus, DepartureTime, Driver}

Given the functional dependency: {Bus}→{Capacity}

To compute {Bus}+, we use reflexivity:

1. **Given**: {Bus}
2. **Reflexivity**: {Bus} includes itself.

So, {Bus}+={Bus, Capacity}

Given the functional dependency: {Route, DepartureTime}→{Bus}

To compute {Route, DepartureTime}+ we use transitivity and reflexivity:

1. **Given**: {Route, DepartureTime}
2. **From FD 3**: {Route, DepartureTime}→{Bus} gives us {Bus}
3. **From FD 2**: {Bus}→{Capacity} gives us {Capacity}
4. **From FD 1**: {Bus, DepartureTime}→{Driver} gives us {Driver}

So, {Route, DepartureTime}+={Route, DepartureTime, Bus, Capacity, Driver}

**Identifying the Key(s) for Relation Shuttle**

From the computed closures,

* {Bus, DepartureTime}+= {Driver} is a transitive dependency
* {Route, DepartureTime}+={Route, DepartureTime, Bus, Capacity, Driver} includes all attributes.

Both closures include all attributes of the relation, indicating that {Route, DepartureTime} can serve as a key for the relation Shuttle, with no partial dependencies.

**Question 5**

Can decompose the initial relation into 2 relations {Bus, Departuretime, Driver, Capacity}, {Route, DepartureTime, Bus}

R1 = {Bus, DepartureTime, Driver{

R2 = {Bus, Capacity}

R3 = {Route, DepartureTime, Bus}

For R1, {Bus, DepartureTime} -> {Driver} so {Bus, DepartureTime would be a superkey and satisfies BCNF.

For R2, {Bus, Capacity}, {Bus} -> {Capacity}, making {Bus} the superkey that satisfies BCNF

For R3, {Route,DepartureTime} -> {Bus}, making {Route, DepartureTime} the key in BCNF

Top of Form