*Deliverable 2*

*Task4*

**Restaurant Delivery System Database Schemas with Keys Underlined**

* Delivery (Did: INTEGER, Vid: INTEGER, DDate: DATE, Dtime:TIME, Is\_delivered :BOOLEAN)

Foreign Key: Vid references Vehicle

* Orders (Order\_id : INTEGER, Cid: INTEGER, Order\_city: VARCHAR(200))

Foreign Key: Cid references Customer

* ComposedBy (Did: INTEGER, Order\_id: INTEGER)

Foreign Key: \_Did\_ references Delivery, Foreign Key: Order\_id references Order

* Customer (Cid: INTEGER, CName: VARCHAR(20),Address: VARCHAR(30), Phone\_Number: VARCHAR(20), Pass\_word: VARCHAR(20), Email: VARCHAR(200))
* Pays (PyId: INTEGER, Order\_id: INTEGER)

Foreign Key: \_PyId\_ references Payment, Foreign Key: \_Order\_id\_ references Order

* Payment (PyId : INTEGER, PDate: DATE)
* Contains (Ordr\_id: INTEGER, Rid: INTEGER, MName: VARCHAR(200))

Foreign Key: Oid references Order, Foreign Key: (Rid, MName) references Meals

* Restaurants (Rid: INTEGER, RName: VARCHAR(20), RCity: VARCHAR(200))
* Meal (Mname: Varchar (20), Rid: Integer, price:REAL)

Foreign Key: \_Rid\_ references Restaurants

* Payment\_Cash (Pcid: INTEGER, Pcprice: REAL, Budget: REAL, Pid: INTEGER,pyid:integer)

Foreign Key: \_Pid\_ references DeliveryPersonal

Foreign Key: \_Pyid\_ references Payment

* PaymentOnline (Poid: INTEGER, POPrice: REAL, Credit\_Card:INTEGER,Ptime:TIME,pyid:Integer)

Foreign Key: \_Pyid\_ references Payment

* Delivery\_Personal (Pid: INTEGER, Vid: INTEGER , Is\_fired:BOOLEAN ,DPCity:VARCHAR(20), PName: VARCHAR(20), DP\_Phone\_Number: VARCHAR(20))

Foreign Key: \_Vid\_ references Vehicle

* Vehicle (Vid: INTEGER, Max\_meals: INTEGER, Speed: REAL, Is\_working:BOOLEAN)

**Explanation of BCNF Compliance for Each Relation in the Restaurant Delivery System Database**

**1.Delivery:**

**Functional Dependency:** {Did}→{DDate,Vid,Dtime,Is\_delivered}

**BCNF Explanation:** Since there are no none trivial functional dependencies , and there is only one Functional dependency and the left hand side is a superkey so the Relation is in BCNF

**2. Order:**

**Functional Dependency:** {order\_id} -> {Cid, Order\_city }

**BCNF Explanation:** The left-hand side {Oid} is a superkey as it uniquely identifies the customer (Cid)and the order city. Therefore, it satisfies the BCNF condition.

**3. ComposedBy:**

**Functional Dependency:** {Did, Order\_id} → {Did, Order\_id}

**BCNF Explanation:** The relation "ComposedBy" has only one functional dependency,

{Did, Order\_id} → {Did, Order\_id}. However, this functional dependency is trivial, as it essentially states that the attributes on the right-hand side are dependent on themselves. Since it does not provide any new information and is considered trivial, the relation is already in BCNF. There is no need for further decomposition, and the design remains in a normalized state.

**4. Customer:**

**Functional Dependency:** {Cid} -> {Address, CPhoneNumber, Password, Email, Cname}, {Email} →{Cid}

**BCNF Explanation:** The left-hand side {Cid} and {Email} are a superkey as they uniquely identifies all the other attributes. Therefore, it satisfies the BCNF condition.

**5. Pays:**

**Functional Dependency:** {PyId, Oid} -> {PyId, Oid}

**BCNF Explanation:** The relation "Pays" has only one functional dependency,

{PyId, Oid} -> {PyId, Oid}.However, this functional dependency is trivial, as it essentially states that the attributes on the right-hand side are dependent on themselves. Since it does not provide any new information and is considered trivial, the relation is already in BCNF. There is no need for further decomposition, and the design remains in a normalized state.

**6. Payment:**

**Functional Dependency:** {PyId} -> {PDate}

**BCNF Explanation:** The left-hand side {PyId} is a superkey as it uniquely identifies the payment date. Therefore, it satisfies the BCNF condition.

**7. Contains:**

**Functional Dependency:** {Oid, {Rid, Mname}} -> {Oid, Rid, MName}

**BCNF Explanation:** The relation "Contains" has only one functional dependency,

{Oid, {Rid, Mname}} -> {Oid, Rid, MName}.

However, this functional dependency is trivial, as it essentially states that the attributes on the right-hand side are dependent on themselves. Since it does not provide any new information and is considered trivial, the relation is already in BCNF. There is no need for further decomposition, and the design remains in a normalized state.

**8. Restaurants:**

**Functional Dependency:** {Rid} -> {RName, RCity}

**BCNF Explanation:** The left-hand side {Rid} is a superkey as it uniquely identifies the restaurant details. Therefore, it satisfies the BCNF condition.

**9. PaymentCash:**

**Functional Dependency:** {Pcid} -> {Pcprice,Pyid, Budget, Pid}

**BCNF Explanation:** The left-hand side {Pcid} is a superkey as it uniquely identifies the details of cash payments. Therefore, it satisfies the BCNF condition.

**10. PaymentOnline:**

**Functional Dependency:** {Poid} -> {POPrice, Pyid,Credit\_card,Ptime}

**BCNF Explanation:** The left-hand side {Poid} is a superkey as it uniquely identifies the details of online payments. Therefore, it satisfies the BCNF condition.

**11. DeliveryPersonal:**

**Functional Dependency:** {Pid} -> {DPCity, Vid, PName, DP\_Phone\_Number,Is\_fired}

**BCNF Explanation:** The left-hand side {Pid} is a superkey as it uniquely identifies the details of delivery personnel. Therefore, it satisfies the BCNF condition.

**12. Vehicle:**

**Functional Dependency:** {Vid} -> {Max\_meals, Speed, Is\_working}

**BCNF Explanation:** The left-hand side {Vid} is a superkey as it uniquely identifies the details of vehicles. Therefore, it satisfies the BCNF condition.

**Summary:**

In each case, the functional dependencies are such that the left-hand side(superkeys) uniquely determines the right-hand side. Therefore, all relations in the schemas are in BCNF.