Part 1: Happy Supplies Parts Warehouse

Step 1: Look at the raw table

Attributes:
customerNumber, customerName, customerType, date, time, employee, partNumber, Name, Type, cageCode, quantityOrdered, unitPrice

Assumption:
- Each physical handwritten form corresponds to one order.
- Any employee can assist any customer, which means that multiple orders from the same customer at different time and date could be assisted by different employees.

order											
customerNumber(PK)	customerName	customerType	**date(PK)**	**time(PK)**	employee	**partNumber(PK)**	partName	partType	cageCode	quantityOrdered	unitPrice
HG54587	Jeff Peterson	consumer	7/1/2024	10:30am	D.Harrison	10654	Float Control	Plumbing	G413	4	12
HG54587	Jeff Peterson	consumer	7/1/2024	10:30am	D.Harrison	10456	Modulator	Electrical	H433	3	7
HG54587	Jeff Peterson	consumer	7/1/2024	10:30am	D.Harrison	10776	Hose Assembly	Plumbing	G413	7	9
HG54587	Jeff Peterson	consumer	7/1/2024	10:30am	D.Harrison	10657	Float Assembly	Plumbing	G413	5	10

Step 2: Define important terms

Primary key (PK):
(customerNumber, date, time, partNumber), because a customer places an order at anytime on any day, containing multiple parts.
Candidate key:
(customerNumber, date, time, partNumber), because a customer places an order at anytime on any day, containing multiple parts.
Other attributes cannot unquiely identify each row.
Functional dependency (FD):
1. customerNumber → customerName, customerType
2. (customerNumber, date, time) → employee(For each order from a customer at a specific time and date is assisted by one employee)
3. partNumber → partName, partType, cageCode, unitPrice
4. (customerNumber, date, time, partNumber) → quantityOrdered (a customer at any time and date can order a certain amount of a part)

Partial dependency:
1. customerName, customerType only dependent on customerNumber
2. employee only dependent on (customerNumber, date, time)
3. partName, partType, cageCode, unitPrice only dependent on partNumber

Transitive dependency:
1. partNumber → partType → cageCode
   a. since parts are categorized and organized by part type that maps to different cage code.

Step 3: Find candidate keys

- Composite key (customerNumber, date, time, partNumber) uniquely identifies a row
  - since a customer can order different parts at multiple times in multiple days; four keys facilitate as a composite key for determination
  - since each of these four is not unique
  - since (date, time) is not unique (different customers can order at the same time)
  - since (customerNumber, partNumber) is not unique (a customers can order same part multiple times)

So: (customerNumber, date, time, partNumber) → primary key.

Step 4: Check Normal Forms

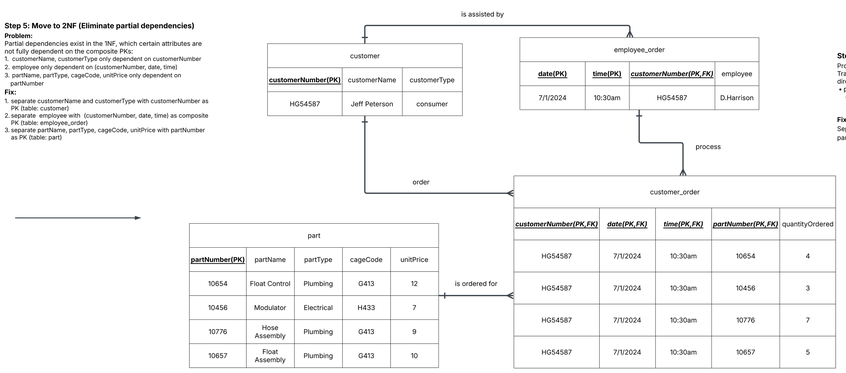
1NF (Eliminate repeating groups)
- Already satisfied: each cell has atomic values (no lists).
- But there are **redundancies**:
  - customerNumber, customerName, customerType, date, time, and employee repeated.
- There are **partial dependencies:**

1. customerName, customerType only dependent on customerNumber
2. employee only dependent on (customerNumber, date, time)
3. partName, partType, cageCode, unitPrice only dependent on partNumber

Step 5: Move to 2NF (Eliminate partial dependencies)

Problem:
Partial dependencies exist in the 1NF, which certain attributes are not fully dependent on the composite PKs:
1. customerName, customerType only dependent on customerNumber
2. employee only dependent on (customerNumber, date, time)
3. partName, partType, cageCode, unitPrice only dependent on partNumber

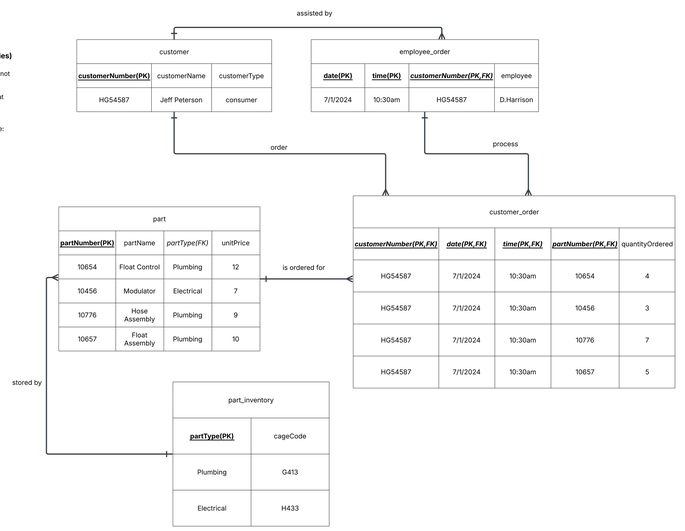
Fix:
1. separate customerName and customerType with customerNumber as PK (table: customer)
2. separate employee with (customerNumber, date, time) as composite PK (table: employee_order)
3. separate partName, partType, cageCode, unitPrice with partNumber as PK (table: part)



Step 6: Move to 3NF (Eliminate transitive dependencies)

Problem:
Transitive dependencies exist in the 2NF, which an attribute is not directly dependent on PK:
- partNumber → partType → cageCode
  - since parts are categorized and organized by part type that maps to different cage code.

Fix:
Separate cageCode to another table with partType as PK (table: part_inventory)



Part 2: Panacea Mental Health Corporation

Step 1: Look at the raw table

Attributes:
staffNo, therapistName, patNo, patName, appointmentDate, appointmentTime, branchNo

Assumption:
Therapists can work at different branches, but at one branch on any given day.
A patient is given an appointment at any time in any day at a branch with a therapist.
Patients can have multiple appointments in a day and with multiple therapists.

therapist_appointment						
staffNo(PK)	therapistName	patNo	patName	**appointmentDate(PK)**	**appointmentTime(PK)**	branchNo
S1011	Fred Smith	P100	Lily White	9/12/2022	10:00	M15
S1011	Fred Smith	P105	Jill Baker	9/12/2022	12:00	M15
S1024	Heidi Pierce	P108	Andy McKee	9/12/2022	10:00	Q10
S1024	Heidi Pierce	P108	Andy McKee	9/14/2022	14:00	Q10
S1032	Richard Levin	P105	Jill Baker	9/14/2022	16:30	M15
S1032	Richard Levin	P110	Jimmy Winter	9/15/2022	18:00	B13

Step 2: Define important terms

Primary key (PK):
- (staffNo, appointmentDate, appointmentTime), as a composite PK
  - It identifies each appointment between a patient and a therapist since a therapist at one branch on a day and can only meet with one patient at one time period.

Candidate key:
- (staffNo, appointmentDate, appointmentTime) can be one
- (staffNo, patNo, appointmentDate, appointmentTime) can be one
  - but that introduces redundancy, since a therapist cannot see two different patients in the same exact time slot.
- (patNo, appointmentDate, appointmentTime) can be one
  - because a patient cannot double-checked appointment with different therapists at the same time; but it does not fully indicate therapist's time slot uniqueness, which is not sufficiently satisfy the demand of company to track therapists.
- Other combinations are not:
  - For example, (appointmentDate, appointmentTime, branchNo) are not valid because at the same time, date and branch, multiple therapists can see multiple different customers.

Functional dependency (FD):
1. staffNo → therapistName
2. (staffNo, appointmentDate) → branchNo (Therapists can work at different branches, but at one branch on any given day.)
3. (staffNo, appointmentDate, appointmentTime) → patNo, patName (A patient is given an appointment at any time in any day at a branch with a therapist.)

Partial dependency:
1. therapistName only dependent on staffNo
2. branchNo only dependent on (staffNo, appointmentDate)
   a. redundancy if identified by the whole PK (staffNo, appointmentDate, appointmentTime) since one therapist work at one branch on any given day.

Transitive dependency:
1. (staffNo, appointmentDate, appointmentTime) → patNo → patName
   a. since patName is dependent on patNo to identify it, it presents the transitive dependency with the composite PK

Step 3: Find candidate keys

- Composite key (staffNo, appointmentDate, appointmentTime) uniquely identifies a row
  - It identifies each appointment between a patient and a therapist since a therapist at one branch on a day and can only meet with one patient at one time period.
  - since each of these four is not unique
- (staffNo, patNo, appointmentDate, appointmentTime) can be one
  - but that introduces redundancy, since a therapist cannot see two different patients in the same exact time slot.
- (patNo, appointmentDate, appointmentTime) can be one
  - but it does not fully indicate therapist's time slot uniqueness, which is not sufficiently satisfy the demand of company to track therapists.
- Other combinations are not:
  - For example, (appointmentDate, appointmentTime, branchNo) are not valid because at the same time, date and branch, multiple therapists can see multiple different customers.

So: (staffNo, appointmentDate, appointmentTime) → primary key.

Step 4: Check Normal Forms

1NF (Eliminate repeating groups)
- Separate appointment day and time as two attributes to satisfy that each cell has atomic values (no lists).
- But there are **redundancies**:
  - staffNo, therapistName, patNo, patName, and branchNo repeated.
- There are **partial dependencies:**
  a. therapistName only dependent on staffNo
  b. branchNo only dependent on (staffNo, appointmentDate)

Step 5: Move to 2NF (Eliminate partial dependencies)

Problem:
Partial dependencies exist in the 1NF, which certain attributes are not fully dependent on the composite PKs:
1. therapistName only dependent on staffNo
2. branchNo only dependent on (staffNo, appointmentDate)
   a. redundancy if identified by the whole PK (staffNo, appointmentDate, appointmentTime) since one therapist work at one branch on any given day.

Fix:
1. separate staffNo and therapistName to another table with staffNo as PK (table: therapist)
2. separate branchNo to another table with (staffNo, appointmentDate) as composite PK (table: therapist_branch_assignment)

Step 6: Move to 3NF (Eliminate transitive dependencies)

Problem:
Transitive dependencies exist in the 2NF, which an attribute is not directly dependent on PK:
- (staffNo, appointmentDate, appointmentTime) → patNo → patName
  - since patName is dependent on patNo to identify it, it presents the transitive dependency with the composite PK

Fix:
Separate patName to another table with patNo as PK (table: patient)

Part 3: Event Management companies

Step 1: Look at the raw table

Attributes:
eNo, contractNo, hours, eName, eventNo, eventLoc

Assumption:
- Each contract only applies to one event.
- An event can have different contracts.
  - therefore eventNo and contractNo is one to many relationship

therapist_appointment					
eNo(PK)	**contractNo(PK)**	hours	eName	eventNo	eventLoc
1135	C1024	16	SmithJ	H25	Queens
1057	C1024	24	HocineD	H25	Queens
1068	C1025	28	WhiteT	H4	Yonkers
1135	C1025	15	SmithJ	H4	Yonkers
1135	C1026	10	SmithJ	H25	Queens

Step 2: Define important terms

Primary key (PK):
- (eNo, contractNo), as a composite PK
  - It identifies each employee's working hours and event that is recorded in different contracts. (most suitable for the business demand)

Candidate key:
- (eNo, contractNo), as a candidate key
- Other combinations are not:
  - (eNo, contractNo, eventNo) can't be because that introduces redundancy, contractNo already identifies eventNo.

Functional dependency (FD):
1. eNo → eName
2. contractNo → eventNo, eventLoc (Each contract only applies to one event.)
3. (eNo, contractNo) → hours

Partial dependency:
1. eName is only dependent on eNo
2. eventNo, eventLoc are dependent on contractNo

Transitive dependency:
1. contractNo → eventNo → eventLoc
   a. eventLoc is dependent on eventNo to identify it, so it present a transitive dependency with contractNo

Step 3: Find candidate keys

- Composite key (eNo, contractNo), as a candidate key
  - It identifies each row
- Other combinations are not:
  - (eNo, contractNo, eventNo) can't be because that creates redundancy, contractNo already identifies eventNo.

So: (eNo, contractNo) → primary key.

Step 4: Check Normal Forms

1NF (Eliminate repeating groups)
- Already satisfied: each cell has atomic values (no lists).
- But there are **redundancies**:
  - eNo, contractNo, *eName*, *eventNo*, and eventLoc repeated.
- There are **partial dependencies:**
  a. eName is only dependent on eNo
  b. eventNo, eventLoc are only dependent on contractNo

Step 5: Move to 2NF (Eliminate partial dependencies)

Problem:
Partial dependencies exist in the 1NF, which certain attributes are not fully dependent on the composite PKs:
1. eName is only dependent on eNo
2. eventNo, eventLoc are only dependent on contractNo

Fix:
1. separate eNo and eName with eNo as PK (table: employee)
2. separate eventNo, eventLoc and contractNo with contractNo as PK (table: contract_event_assignment)

Step 6: Move to 3NF (Eliminate transitive dependencies)

Problem:
Transitive dependencies exist in the 2NF, which an attribute is not directly dependent on PK:
1. contractNo → eventNo → eventLoc
   a. eventLoc is dependent on eventNo to identify it, so it present a transitive dependency with contractNo

Fix:
Separate eventLoc to another table with eventNo as PK (table: event)