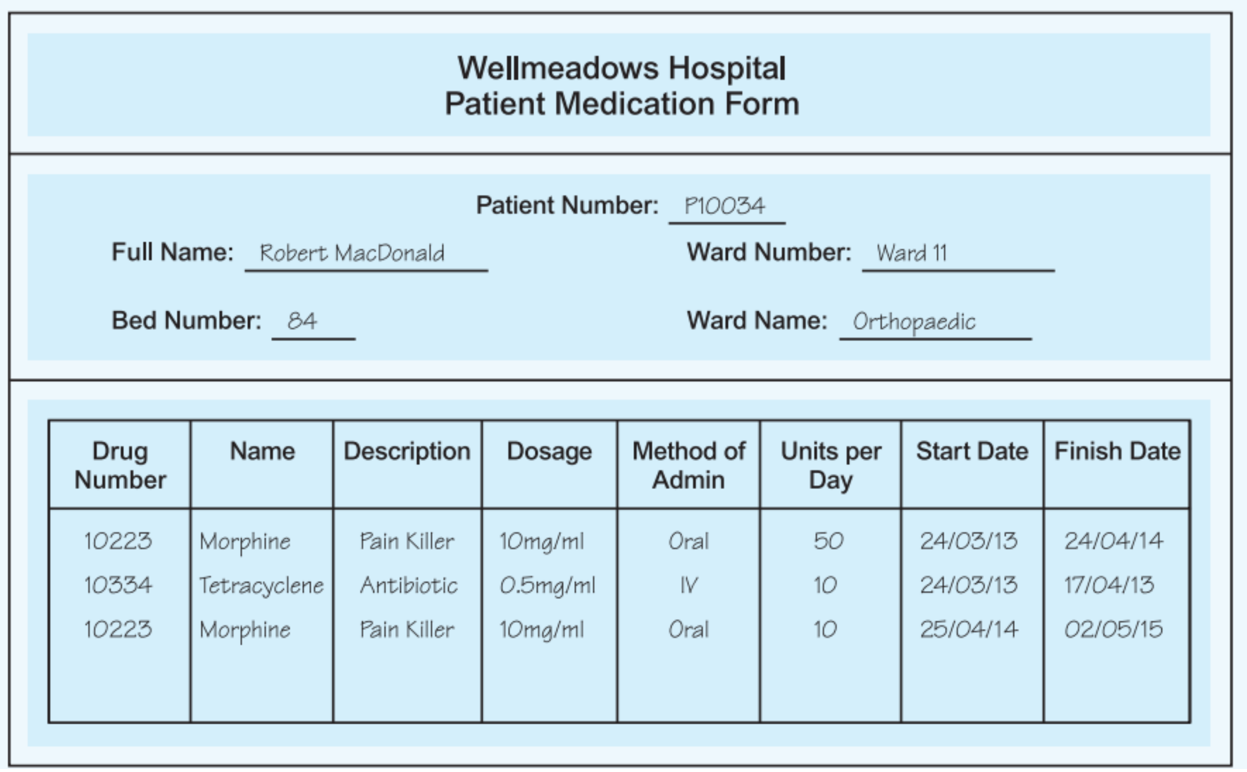
QUESTION NO1: Examine the Patient Medication Form for the Wellmeadows Hospital case study (see Appendix B) shown in Figure 14.18.

(a) Identify the functional dependencies represented by the attributes shown in the form in Figure 14.18. State any assumptions that you make about the data and the attributes shown in this form.

(b) Describe and illustrate the process of normalizing the attributes shown in Figure 14.18 to produce a set of well-designed 3NF relations.

(c) Identify the primary, alternate, and foreign keys in your 3NF relations.



Wellmeadows Hospital
Patient Medication Form

Patient Number: P10034

Full Name: Robert MacDonald **Ward Number:** Ward 11

Bed Number: 84 **Ward Name:** Orthopaedic

Drug Number	Name	Description	Dosage	Method of Admin	Units per Day	Start Date	Finish Date
10223	Morphine	Pain Killer	10mg/ml	Oral	50	24/03/13	24/04/14
10334	Tetracyclene	Antibiotic	0.5mg/ml	IV	10	24/03/13	17/04/13
10223	Morphine	Pain Killer	10mg/ml	Oral	10	25/04/14	02/05/15

(a) Identify Functional Dependencies

To identify functional dependencies, let's analyze the form:

Key observations:

1. **Patient details** (Patient Number, Full Name, Ward Number, Ward Name, Bed Number) uniquely identify a patient.
2. **Drug details** (Drug Number, Name, Description, Dosage, Method of Admin, Units per Day, Start Date, Finish Date) describe medication prescribed to a patient.
3. Each drug prescription is uniquely identified by the combination of **Patient Number** and **Drug Number**.

Functional Dependencies (FDs):

1. **Patient Number → Full Name, Ward Number, Ward Name, Bed Number**
   - A patient is uniquely identified by their Patient Number.
2. **Drug Number → Name, Description, Dosage**
   - A drug is uniquely identified by its Drug Number.
3. **Patient Number, Drug Number → Method of Admin, Units per Day, Start Date, Finish Date**

  - A prescription is uniquely identified by a combination of Patient Number and Drug Number.

(b) Normalization Process to 3NF

We'll normalize the table step by step:

Unnormalized Form (UNF):

The given table mixes patient, ward, and drug data. It contains redundancy, such as repeated patient details and drug descriptions.

First Normal Form (1NF):

To achieve 1NF:

- Remove duplicate and repeating data.
- Ensure every column contains atomic (indivisible) values.

1NF Relation:

Patient Number	Full Name	Ward Number	Ward Name	Bed Number	Drug Number	Name	Description	Dosage	Method of Admin	Units per Day	Start Date	Finish Date
P10034	Robert MacDonald	Ward 11	Orthopaedic	84	10223	Morphine	Pain Killer	10mg/ml	Oral	50	24/03/13	24/04/14
P10034	Robert MacDonald	Ward 11	Orthopaedic	84	10334	Tetracycline	Antibiotic	0.5mg/ml	IV	10	24/03/13	17/04/13
P10034	Robert MacDonald	Ward 11	Orthopaedic	84	10223	Morphine	Pain Killer	10mg/ml	Oral	10	25/04/14	02/05/15

Second Normal Form (2NF):

To achieve 2NF:

- Remove partial dependencies (where non-prime attributes depend only on part of a composite primary key).

Step 1: Identify Primary Key: The composite primary key is **(Patient Number, Drug Number)**.

Step 2: Remove Partial Dependencies:

- **Full Name, Ward Number, Ward Name, Bed Number** depend only on **Patient Number**. Create a separate table for patient data.
- **Name, Description, Dosage** depend only on **Drug Number**. Create a separate table for drug data.

2NF Relations:

1. **Patient Table:**

Patient Number	Full Name	Ward Number	Ward Name	Bed Number
P10034	Robert MacDonald	Ward 11	Orthopaedic	84

2. **Drug Table:**

Drug Number	Name	Description	Dosage
10223	Morphine	Pain Killer	10mg/ml
10334	Tetracycline	Antibiotic	0.5mg/ml

3. **Prescription Table:**

Patient Number	Drug Number	Method of Admin	Units per Day	Start Date	Finish Date
P10034	10223	Oral	50	24/03/13	24/04/14
P10034	10334	IV	10	24/03/13	17/04/13
P10034	10223	Oral	10	25/04/14	02/05/15

Third Normal Form (3NF):

To achieve 3NF:

- Remove transitive dependencies (non-prime attributes depending on other non-prime attributes).

In this case, there are no transitive dependencies left. The relations are already in 3NF.

(c) Identify Keys

Primary Keys:

1. **Patient Table:** Patient Number
2. **Drug Table:** Drug Number
3. **Prescription Table:** (Patient Number, Drug Number)

Alternate Keys: None in this case.

Foreign Keys:

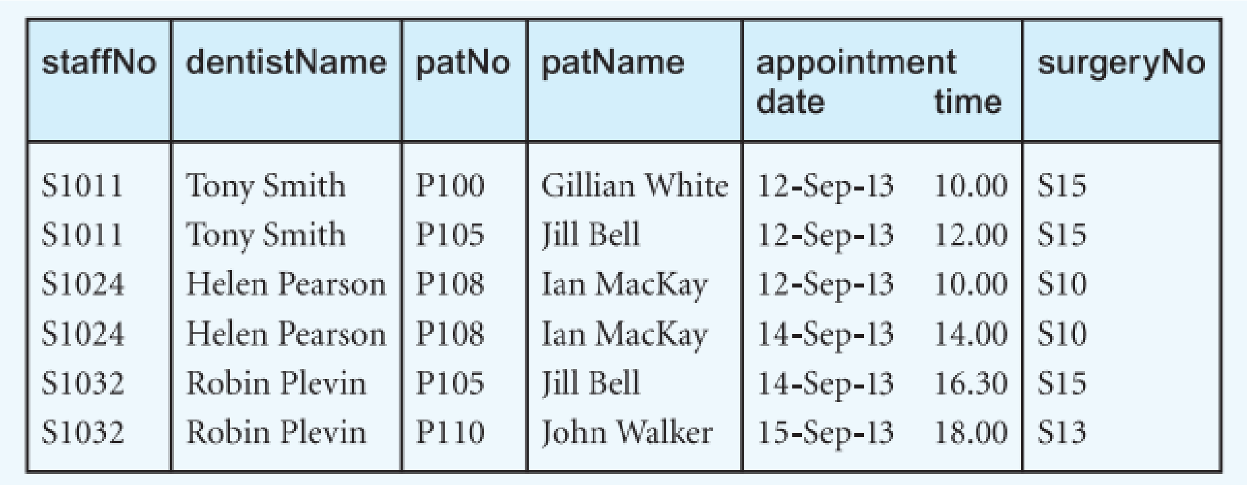
1. **Prescription Table:**
   - Patient Number (references Patient Table)
   - Drug Number (references Drug Table)

QUESTION 2: The table shown in Figure 14.19 lists sample dentist/patient appointment data. A patient is given an appointment at a specific time and date with a dentist located at a particular surgery. On each day of patient appointments, a dentist is allocated to a specific surgery for that day.

(a) The table shown in Figure 14.19 is susceptible to update anomalies. Provide examples of insertion, deletion, and update anomalies.

(b) Identify the functional dependencies represented by the attributes shown in the table of Figure 14.19. State any assumptions you make about the data and the attributes shown in this table.

(c) Describe and illustrate the process of normalizing the table shown in Figure 14.19 to 3NF relations. Identify the primary, alternate, and foreign keys in your 3NF relations.



staffNo	dentistName	patNo	patName	appointment		surgeryNo
				date	time	
S1011	Tony Smith	P100	Gillian White	12-Sep-13	10.00	S15
S1011	Tony Smith	P105	Jill Bell	12-Sep-13	12.00	S15
S1024	Helen Pearson	P108	Ian MacKay	12-Sep-13	10.00	S10
S1024	Helen Pearson	P108	Ian MacKay	14-Sep-13	14.00	S10
S1032	Robin Plevin	P105	Jill Bell	14-Sep-13	16.30	S15
S1032	Robin Plevin	P110	John Walker	15-Sep-13	18.00	S13

(a) Examples of Update Anomalies

1. Insertion Anomaly:

- If a new patient is added but hasn't scheduled an appointment, their information (e.g., `patNo` and `patName`) cannot be recorded without filling out unrelated attributes like `appointment date`, `time`, `staffNo`, etc.
- Example: We can't add a new patient John Smith (P111) unless an appointment is also scheduled.

2. Deletion Anomaly:

- Deleting an appointment may result in the loss of critical information about a patient or dentist.
- Example: If the only appointment for Ian MacKay (P108) on 14-Sep-13 is deleted, all information about Ian MacKay is lost.

3. Update Anomaly:

- If a dentist or patient's information changes, it has to be updated in multiple rows, leading to inconsistencies.
- Example: If Tony Smith (`staffNo S1011`) moves to a new surgery, every row with Tony Smith must be updated. If one row is missed, it creates inconsistency.

(b) Functional Dependencies

Assumptions:

1. Each dentist is identified by a unique `staffNo`.
2. Each patient is identified by a unique `patNo`.
3. A dentist works in one surgery per day.
4. A patient has only one appointment at a specific date and time.

Functional Dependencies:

1. **staffNo → dentistName**
   - A dentist's name is uniquely identified by their staff number.
2. **patNo → patName**
   - A patient's name is uniquely identified by their patient number.
3. **staffNo, appointment date → surgeryNo**
   - On any given day, a dentist is assigned to one surgery.
4. **patNo, appointment date, appointment time → staffNo, surgeryNo**
   - A specific patient appointment determines the dentist and surgery.

(c) Normalization Process to 3NF

Unnormalized Form (UNF):

The given table mixes patient, dentist, appointment, and surgery data.

First Normal Form (1NF):

To achieve 1NF:

- Ensure each column contains atomic values.

The current table already satisfies 1NF.

Second Normal Form (2NF):

To achieve 2NF:

- Remove partial dependencies (attributes depending only on part of the composite primary key).

Primary Key: (`patNo`, `appointment date`, `appointment time`)

Step 1: Remove Partial Dependencies

- `staffNo → dentistName`: Move dentist information to a separate table.
- `patNo → patName`: Move patient information to a separate table.

2NF Relations:

1. **Patient Table:**

patNo	**patName**
P100	Gillian White
P105	Jill Bell
P108	Ian MacKay
P110	John Walker

2. **Dentist Table:**

staffNo	**dentistName**
S1011	Tony Smith
S1024	Helen Pearson
S1032	Robin Plevin

3. **Appointment Table:**

patNo	**appointment date**	**appointment time**	**staffNo**	**surgeryNo**
P100	12-Sep-13	10.00	S1011	S15
P105	12-Sep-13	12.00	S1011	S15
P108	12-Sep-13	10.00	S1024	S10
P108	14-Sep-13	14.00	S1024	S10
P105	14-Sep-13	16.30	S1032	S15
P110	15-Sep-13	18.00	S1032	S13

Third Normal Form (3NF):

To achieve 3NF:

- Remove transitive dependencies (non-prime attributes depending on other non-prime attributes).

Step 1: Remove Transitive Dependencies

- `staffNo, appointment date → surgeryNo`: Move surgery allocation to a separate table.

3NF Relations:

1. **Patient Table:**

patNo	**patName**
P100	Gillian White
P105	Jill Bell
P108	Ian MacKay
P110	John Walker

2. **Dentist Table:**

staffNo	**dentistName**
S1011	Tony Smith
S1024	Helen Pearson

S1032	Robin Plevin

3. **Surgery Allocation Table:**

staffNo	appointment date	surgeryNo
S1011	12-Sep-13	S15
S1024	12-Sep-13	S10
S1024	14-Sep-13	S10
S1032	14-Sep-13	S15
S1032	15-Sep-13	S13

4. **Appointment Table:**

patNo	appointment date	appointment time	staffNo
P100	12-Sep-13	10.00	S1011
P105	12-Sep-13	12.00	S1011
P108	12-Sep-13	10.00	S1024
P108	14-Sep-13	14.00	S1024
P105	14-Sep-13	16.30	S1032
P110	15-Sep-13	18.00	S1032

Keys in 3NF Relations

Primary Keys:

1. Patient Table: **patNo**
2. Dentist Table: **staffNo**
3. Surgery Allocation Table: **(staffNo, appointment date)**
4. Appointment Table: **(patNo, appointment date, appointment time)**

Alternate Keys:

None in this case.

Foreign Keys:

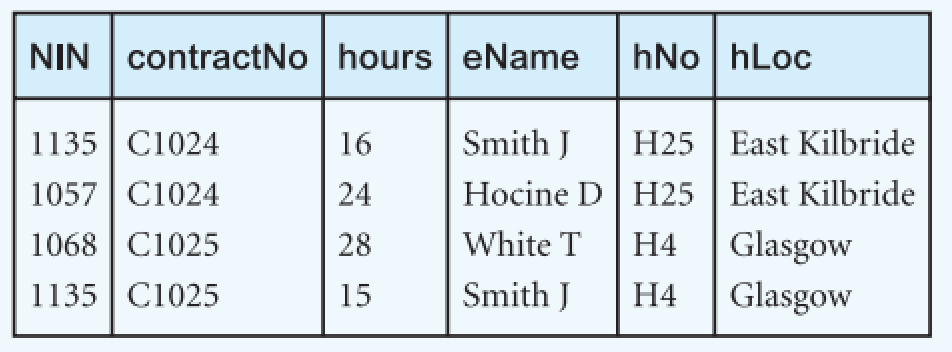
1. Appointment Table:
   - `patNo` → Patient Table
   - `staffNo` → Dentist Table
   - `(staffNo, appointment date)` → Surgery Allocation Table

QUESTION NO 3: An agency called Instant Cover supplies part-time/temporary staff to hotels within Scotland. The table shown in Figure 14.20 displays sample data, which lists the time spent by agency staff working at various hotels. The National Insurance Number (NIN) is unique for every member of staff.

(a) The table shown in Figure 14.20 is susceptible to update anomalies. Provide examples of insertion, deletion, and update anomalies.

(b) Identify the functional dependencies represented by the attributes shown in the table of Figure 14.20. State any assumptions that you make about the data and the attributes shown in this table.

(c) Describe and illustrate the process of normalizing the table shown in Figure 14.20 to 3NF. Identify primary, alternate, and foreign keys in your relations.



NIN	contractNo	hours	eName	hNo	hLoc
1135	C1024	16	Smith J	H25	East Kilbride
1057	C1024	24	Hocine D	H25	East Kilbride
1068	C1025	28	White T	H4	Glasgow
1135	C1025	15	Smith J	H4	Glasgow

(a) Examples of Update Anomalies

1. Insertion Anomaly:

- If a new hotel needs to be added but no staff is assigned yet, the hotel's details (e.g., `hNo` and `hLoc`) cannot be recorded without adding unrelated data like `NIN` or `hours`.
- Example: A new hotel in Edinburgh (H20) cannot be added unless a staff member is also linked to it.

2. Deletion Anomaly:

- Deleting a staff record may result in the loss of hotel details.
- Example: If all assignments for H25 (East Kilbride) are deleted, the information about H25 and its location is also lost.

3. Update Anomaly:

- If a hotel's location changes, it must be updated in multiple rows, leading to inconsistencies.
- Example: If H4 changes its location from Glasgow to Edinburgh, every row with H4 must be updated. If one row is missed, it creates inconsistent data.

(b) Functional Dependencies

Assumptions:

1. Each staff member is uniquely identified by `NIN`.
2. Each hotel is uniquely identified by `hNo`.
3. Each contract is uniquely identified by `contractNo`.
4. `hNo` determines `hLoc` (a hotel's location).
5. `NIN` determines `eName` (staff name).
6. A combination of `NIN` and `contractNo` determines `hours` worked.

Functional Dependencies:

1. **NIN → eName**
   - A staff member’s name is uniquely identified by their NIN.
2. **hNo → hLoc**
   - A hotel’s location is uniquely determined by its hotel number.
3. **contractNo → (hNo, hLoc)**
   - A contract is linked to a specific hotel.
4. **NIN, contractNo → hours**
   - The hours worked by a staff member are determined by the combination of their NIN and the contract.

(c) Normalization Process to 3NF

Unnormalized Form (UNF):

The given table combines staff, hotel, and contract data.

First Normal Form (1NF):

The table already satisfies 1NF as all attributes contain atomic values.

Second Normal Form (2NF):

To achieve 2NF:

- Remove partial dependencies (attributes depending only on part of the composite primary key).

Primary Key (Candidate): (`NIN`, `contractNo`)

Step 1: Remove Partial Dependencies

- `NIN → eName`: Move staff details to a separate table.
- `hNo → hLoc`: Move hotel details to a separate table.
- `contractNo → hNo, hLoc`: Move contract details to a separate table.

2NF Relations:

1. **Staff Table:**

NIN	**eName**
1135	Smith J
1057	Hocine D
1068	White T

2. **Hotel Table:**

hNo	**hLoc**
H25	East Kilbride
H4	Glasgow

3. **Contract Table:**

contractNo	**hNo**
C1024	H25
C1025	H4

4. **Assignment Table:**

NIN	**contractNo**	**hours**
1135	C1024	16
1057	C1024	24
1068	C1025	28
1135	C1025	15

Third Normal Form (3NF):

To achieve 3NF:

- Remove transitive dependencies (non-prime attributes depending on other non-prime attributes).

Step 1: Remove Transitive Dependencies

- There are no transitive dependencies in this case, as all non-prime attributes directly depend on their respective keys.

3NF Relations: The relations derived in 2NF are already in 3NF.

Keys in 3NF Relations

Primary Keys:

1. **Staff Table: NIN**
2. **Hotel Table: hNo**
3. **Contract Table: contractNo**
4. **Assignment Table: (NIN, contractNo)**

Alternate Keys:

None in this case.

Foreign Keys:

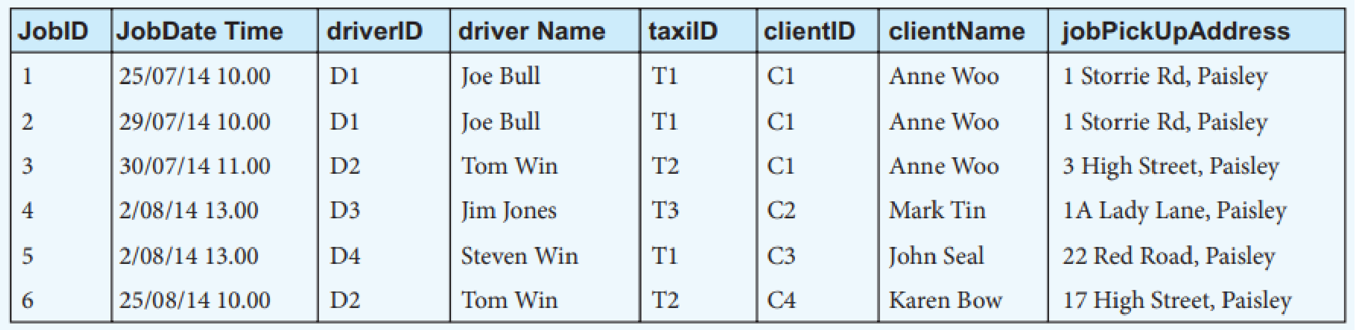
1. **Contract Table:**
   - `hNo` → Hotel Table
2. **Assignment Table:**
   - `NIN` → Staff Table
   - `contractNo` → Contract Table

QUESTION NO 4: A company called FastCabs provides a taxi service to clients. The table shown in Figure 14.21 displays some details of client bookings for taxis. Assume that a taxi driver is assigned to a single taxi, but a taxi can be assigned to one or more drivers.

(a) Identify the functional dependencies that exist between the columns of the table in Figure 14.21 and identify the primary key and any alternate key(s) (if present) for the table.

(b) Describe why the table in Figure 14.21 is not in 3NF.

(c) The table shown in Figure 14.21 is susceptible to update anomalies. Provide examples of how insertion, deletion, and modification anomalies could occur on this table.



JobID	JobDate Time	driverID	driver Name	taxiID	clientID	clientName	jobPickUpAddress
1	25/07/14 10.00	D1	Joe Bull	T1	C1	Anne Woo	1 Storrie Rd, Paisley
2	29/07/14 10.00	D1	Joe Bull	T1	C1	Anne Woo	1 Storrie Rd, Paisley
3	30/07/14 11.00	D2	Tom Win	T2	C1	Anne Woo	3 High Street, Paisley
4	2/08/14 13.00	D3	Jim Jones	T3	C2	Mark Tin	1A Lady Lane, Paisley
5	2/08/14 13.00	D4	Steven Win	T1	C3	John Seal	22 Red Road, Paisley
6	25/08/14 10.00	D2	Tom Win	T2	C4	Karen Bow	17 High Street, Paisley

Step 1: First Normal Form (1NF)

Definition: A table is in **1NF** if:

1. It has no repeating groups or multivalued attributes.
2. Each cell contains a single value, and each column contains atomic data.

The given table is already in **1NF** because:

- All attributes contain atomic values.
- There are no repeating groups.

Step 2: Second Normal Form (2NF)

Definition: A table is in **2NF** if:

1. It is in **1NF**.
2. It has no partial dependencies (non-prime attributes must depend on the whole primary key, not just part of it).

Issues in the Table:

- `driver Name`, `clientName`, and `jobPickUpAddress` depend only on `driverID` and `clientID`, not on the whole primary key (`JobID`).

Solution: Decompose the table into two or more tables to eliminate partial dependencies.

Decomposed Tables in 2NF:

1. **Job Table (Main Table)**:
   - Attributes: `JobID`, `JobDate Time`, `driverID`, `taxiID`, `clientID`
   - Primary Key: `JobID`
   - Description: This table stores job details.

JobID	JobDate Time	driverID	taxiID	clientID
1	25/07/14 10.00	D1	T1	C1
2	29/07/14 10.00	D1	T1	C1
3	30/07/14 11.00	D2	T2	C1
4	2/08/14 13.00	D3	T3	C2
5	2/08/14 13.00	D4	T1	C3
6	25/07/14 10.00	D2	T2	C4

2. **Driver Table**:
   - Attributes: `driverID`, `driver Name`, `taxiID`
   - Primary Key: `driverID`
   - Description: This table stores driver details.

driverID	driver Name	taxiID
D1	Joe Bull	T1
D2	Tom Win	T2
D3	Jim Jones	T3
D4	Steven Win	T1

3. **Client Table**:
   - Attributes: `clientID`, `clientName`, `jobPickUpAddress`
   - Primary Key: `clientID`
   - Description: This table stores client details.

clientID	clientName	jobPickUpAddress
C1	Anne Woo	1 Storrie Rd, Paisley
C2	Mark Tin	1A Lady Lane, Paisley
C3	John Seal	22 Red Road, Paisley
C4	Karen Bow	17 High Street, Paisley

Step 3: Third Normal Form (3NF)

Definition: A table is in **3NF** if:

1. It is in **2NF**.
2. There are no transitive dependencies (non-prime attributes depending on non-key attributes).

Issues in 2NF:

- In the **Driver Table**, `taxiID` determines `driverID` and vice versa. This creates a transitive dependency.

Solution: Further decompose the Driver Table.

Final Tables in 3NF:

1. **Job Table** (same as 2NF):

JobID	JobDate Time	driverID	taxiID	clientID
1	25/07/14 10.00	D1	T1	C1
2	29/07/14 10.00	D1	T1	C1
3	30/07/14 11.00	D2	T2	C1
4	2/08/14 13.00	D3	T3	C2
5	2/08/14 13.00	D4	T1	C3
6	25/07/14 10.00	D2	T2	C4

2. **Driver Table**:
   - Attributes: `driverID`, `driver Name`
   - Primary Key: `driverID`

driverID	driver Name
D1	Joe Bull
D2	Tom Win
D3	Jim Jones
D4	Steven Win

3. **Taxi Table**:
   - Attributes: `taxiID`, `driverID`
   - Primary Key: `taxiID`

taxiID	driverID
T1	D1
T2	D2
T3	D3
T1	D4

4. **Client Table** (same as 2NF):

clientID	clientName	jobPickUpAddress
C1	Anne Woo	1 Storrie Rd, Paisley
C2	Mark Tin	1A Lady Lane, Paisley
C3	John Seal	22 Red Road, Paisley
C4	Karen Bow	17 High Street, Paisley

Summary:

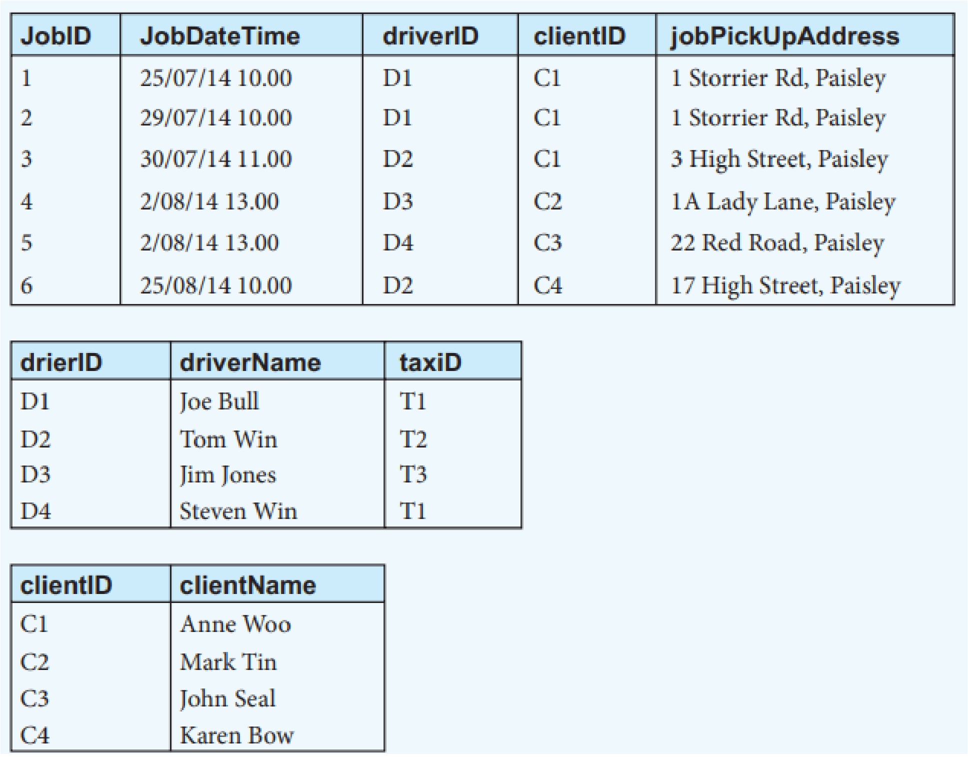
1. The original table is split into **four smaller tables**: Job Table, Driver Table, Taxi Table, and Client Table.
2. All tables are now in **3NF** with no partial or transitive dependencies.
3. This structure eliminates redundancy and minimizes update anomalies.

QUESTION NO 5: Applying normalisation to 3NF on the table shown in Figure 14.21 results in the formation of the three 3NF tables shown in Figure 14.22.

(a) Identify the functional dependencies that exist between the columns of each table in Figure 14.22 and identify the primary key and any alternate and foreign key(s) (if present) for each table.

(b) Describe why storing the FastCabs data across three 3NF tables avoids the update anomalies described in Exercise 14.17(b).

(c) Describe how the original table shown in Figure 14.21 can be re-created through relational joins between primary key and foreign keys columns of the tables in Figure 14.22.



JobID	JobDateTime	driverID	clientID	jobPickUpAddress
1	25/07/14 10.00	D1	C1	1 Storrier Rd, Paisley
2	29/07/14 10.00	D1	C1	1 Storrier Rd, Paisley
3	30/07/14 11.00	D2	C1	3 High Street, Paisley
4	2/08/14 13.00	D3	C2	1A Lady Lane, Paisley
5	2/08/14 13.00	D4	C3	22 Red Road, Paisley
6	25/08/14 10.00	D2	C4	17 High Street, Paisley

drierID	driverName	taxiD
D1	Joe Bull	T1
D2	Tom Win	T2
D3	Jim Jones	T3
D4	Steven Win	T1

clientID	clientName
C1	Anne Woo
C2	Mark Tin
C3	John Seal
C4	Karen Bow

(a) Functional Dependencies, Primary Key, and Foreign Keys

1. Job Table (`JobID`, `JobDateTime`, `driverID`, `clientID`, `jobPickUpAddress`):

- **Functional Dependencies**:
    1. `JobID → JobDateTime, driverID, clientID, jobPickUpAddress`
        - `JobID` uniquely determines the rest of the columns in this table.
- **Primary Key**: `JobID`
    - Uniquely identifies each row in the Job table.
- **Foreign Keys**:
    0. `driverID` references the `driverID` in the **Driver Table**.
    1. `clientID` references the `clientID` in the **Client Table**.

---

2. Driver Table (`driverID`, `driverName`, `taxiID`):

- **Functional Dependencies**:
    1. `driverID → driverName, taxiID`
        - Each `driverID` determines the driver's name and the assigned taxi.
- **Primary Key**: `driverID`
    - Uniquely identifies each row in the Driver table.
- **Foreign Key**: None in this table.

---

3. Client Table (`clientID`, `clientName`):

- **Functional Dependencies**:
    1. `clientID → clientName`
        - `clientID` determines the client's name.
- **Primary Key**: `clientID`
    - Uniquely identifies each row in the Client table.
- **Foreign Key**: None in this table.

(b) How storing data in 3NF tables avoids update anomalies

1. **Insertion Anomaly**:
    - In the original table, adding a new driver or client required adding a new job, even if no job exists.
    - In the 3NF structure, new drivers can be added to the **Driver Table**, and new clients can be added to the **Client Table**, independently of jobs.
2. **Deletion Anomaly**:
    - In the original table, deleting a job could result in the loss of driver or client details.
    - In the 3NF structure, deleting a job from the **Job Table** does not affect the **Driver Table** or **Client Table**, preserving driver and client details.
3. **Modification Anomaly**:
    - In the original table, changing a driver's name or client's address required updating multiple rows, risking inconsistent data.
    - In the 3NF structure, driver details are updated in the **Driver Table**, and client details are updated in the **Client Table**, avoiding redundancy and inconsistency.