Written Final

Step-by-Step Process

Step 1: Describe the Background Story of the Business

[REDACTED] Inc.

[REDACTED] Inc. offers specialized redaction services to ensure the privacy and security of sensitive documents and images for clients. This business primarily serves legal entities needing to redact sensitive information from legal documents, companies complying with privacy laws like GDPR and HIPAA, and individuals seeking to anonymize personal information before sharing.

The key services include text redaction, image obfuscation (eg blurring license plates), and customized redactions based on client requirements. The core business processes include client document submission, the redaction process, client reviews, and the invoicing and payment for services rendered.

The company doesn't take itself too seriously, having a redacted ice cream cone for a logo.

Step 2: Make Assumptions and Create the ER Model

I have made the following assumptions:

1. Each client can submit multiple documents.
2. Each document can undergo multiple redactions.
3. Each document can be reviewed multiple times before final approval.
4. Each client receives an invoice for each redaction project, which can include multiple line items.
5. Each invoice will have at least one invoice item.
6. Payments are made per invoice.

Based on these assumptions, I will include entities of Clients, Documents, Redactions, Reviews, Invoices, InvoiceItems, and Payments into the ER model. At this point I have a rough idea of needed attributes, as well as cardinality and participation.

In general, I will always attempt to have a single primary key per entity from the start, splitting off attributes into additional entities (ex InvoiceItems from Invoices) as I go, to keep a single responsibility per table. While this step is supposed to be only conceptual, by forcing single PK entities, the FD1 on each entity comes for free.

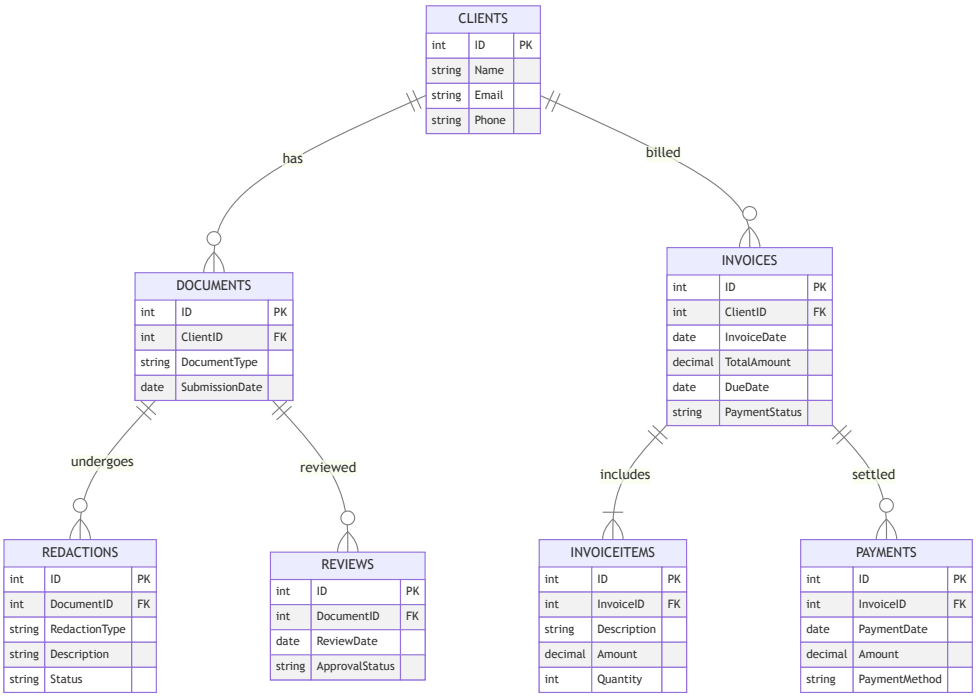
```
CLIENTS(ID (PK), Name, Email, Phone)
        FD1: ID -> Name, Email, Phone

DOCUMENTS(ID (PK), ClientID (FK), DocumentType, SubmissionDate)
        FD1: ID -> ClientID, DocumentType, SubmissionDate

REDACTIONS(ID (PK), DocumentID (FK), RedactionType, Description, Status)
        FD1: ID -> DocumentID, RedactionType, Description, Status

REVIEWS(ID (PK), DocumentID (FK), ReviewDate, ApprovalStatus)
        FD1: ID -> DocumentID, ReviewDate, ApprovalStatus

INVOICES(ID (PK), ClientID (FK), InvoiceDate, TotalAmount, DueDate, PaymentStatus)
        FD1: ID -> ClientID, InvoiceDate, TotalAmount, DueDate, PaymentStatus

INVOICEITEMS(ID (PK), InvoiceID (FK), Description, Amount, Quantity)
        FD1: ID -> InvoiceID, Description, Amount, Quantity

PAYMENTS(ID (PK), InvoiceID (FK), PaymentDate, Amount, PaymentMethod)
        FD1: ID -> InvoiceID, PaymentDate, Amount, PaymentMethod
```

Note on ER formatting: I don't like the underline to represent primary keys and so use the (PK) as needed, and in the remainder of this write up.

```
CLIENT_DOCUMENTS
        CLIENTS.ID -> DOCUMENTS.ClientID
        Cardinality: One CLIENTS can have many DOCUMENTS

DOCUMENT_REDACTIONS
        DOCUMENTS.ID -> REDACTIONS.DocumentID
        Cardinality: One DOCUMENT can have many REDACTIONS

DOCUMENT_REVIEWS
        DOCUMENTS.ID -> REVIEWS.DocumentID
        Cardinality: One DOCUMENT can have many REVIEWS

CLIENT_INVOICES
        CLIENTS.ID -> INVOICES.ClientID
        Cardinality: One CLIENTS can have many INVOICES

INVOICE_INVOICEITEMS
        INVOICES.ID -> INVOICEITEMS.InvoiceID
        Cardinality: One INVOICES can have many INVOICEITEMS

INVOICE_PAYMENTS
        INVOICES.ID -> PAYMENTS.InvoiceID
        Cardinality: One INVOICES can have many PAYMENTS
```

Step 3: Create the Entity-Relationship Diagram (ERD)

For this step I utilized mermaid (mermaidchart.com) to visualize the crow's feet syntax. I noticed that step 2 and 3, for me, are more iterative than a linear progression -- as I added more assumptions and corrected my cardinality/participation understandings. For example in the first iteration I had Invoices that could contain zero InvoiceItems.




```
erDiagram

    CLIENTS ||--o{ DOCUMENTS : "has"
    CLIENTS {
        int ID PK
        string Name
        string Email
        string Phone
    }

    DOCUMENTS ||--o{ REDACTIONS : "undergoes"
    DOCUMENTS ||--o{ REVIEWS : "reviewed"
    DOCUMENTS {
        int ID PK
        int ClientID FK
        string DocumentType
        date SubmissionDate
```

```
    }

    REDACTIONS {
        int ID PK
        int DocumentID FK
        string RedactionType
        string Description
        string Status
    }

    REVIEWS {
        int ID PK
        int DocumentID FK
        date ReviewDate
        string ApprovalStatus
    }

    CLIENTS ||--o{ INVOICES : "billed"
    INVOICES ||--|{ INVOICEITEMS : "includes"
    INVOICES ||--o{ PAYMENTS : "settled"
    INVOICES {
        int ID PK
        int ClientID FK
        date InvoiceDate
        decimal TotalAmount
        date DueDate
        string PaymentStatus
    }

    INVOICEITEMS {
        int ID PK
        int InvoiceID FK
        string Description
        decimal Amount
        int Quantity
    }

    PAYMENTS {
        int ID PK
        int InvoiceID FK
        date PaymentDate
        decimal Amount
        string PaymentMethod
    }
```

Step 4: Convert the ERD to a Relational Model

From the ERD, the updates were to cardinality and participation, more than structure of the RM (from ER)

```
CLIENTS(ID (PK), Name, Email, Phone)
        FD: ID -> Name, Email, Phone

DOCUMENTS(ID (PK), ClientID (FK), DocumentType, SubmissionDate)
        FD: ID -> ClientID, DocumentType, SubmissionDate

REDACTIONS(ID (PK), DocumentID (FK), RedactionType, Description, Status)
        FD: ID -> DocumentID, RedactionType, Description, Status

REVIEWS(ID (PK), DocumentID (FK), ReviewDate, ApprovalStatus)
        FD: ID -> DocumentID, ReviewDate, ApprovalStatus

INVOICES(ID (PK), ClientID (FK), InvoiceDate, TotalAmount, DueDate, PaymentStatus)
        FD: ID -> ClientID, InvoiceDate, TotalAmount, DueDate, PaymentStatus

INVOICEITEMS(ID (PK), InvoiceID (FK), Description, Amount, Quantity)
        FD: ID -> InvoiceID, Description, Amount, Quantity

PAYMENTS(ID (PK), InvoiceID (FK), PaymentDate, Amount, PaymentMethod)
        FD: ID -> InvoiceID, PaymentDate, Amount, PaymentMethod
```

```
CLIENTS and DOCUMENTS:
        Relationship: "has"
        Cardinality: One CLIENT can have zero or many DOCUMENTS.
                                Each DOCUMENT is associated with exactly one CLIENT.
        Participation: Optional for DOCUMENTS and Mandatory for CLIENTS.

DOCUMENTS and REDACTIONS:
        Relationship: "undergoes"
        Cardinality: One DOCUMENT can have zero or many REDACTIONS.
                                Each REDACTION is associated with exactly one DOCUMENT.
```

```
        Participation: Optional for REDACTIONS and Mandatory for DOCUMENTS.

DOCUMENTS and REVIEWS:
        Relationship: "reviewed"
        Cardinality: One DOCUMENT can be reviewed zero or many times.
                                Each REVIEW is associated with exactly one DOCUMENT.
        Participation: Optional for REVIEWS and Mandatory for DOCUMENTS.

CLIENTS and INVOICES:
        Relationship: "billed"
        Cardinality: One CLIENT can have zero or many INVOICES.
                                Each INVOICE is associated with exactly one CLIENT.
        Participation: Optional for INVOICES and Mandatory for CLIENTS.

INVOICES and INVOICEITEMS:
        Relationship: "includes"
        Cardinality: One INVOICE can include one or many INVOICEITEMS.
                                Each INVOICEITEM is associated with exactly one INVOICE.
        Participation: Mandatory for both INVOICES and INVOICEITEMS.

INVOICES and PAYMENTS:
        Relationship: "settled"
        Cardinality: One INVOICE can be settled by zero or many PAYMENTS.
                                Each PAYMENT is associated with exactly one INVOICE.
        Participation: Optional for PAYMENTS and Mandatory for INVOICES.
```

Step 5: Normalize the Relational Model to 3NF

For normalization I verify all entities, in order of ascending normalized forms:

1NF

- Every attribute is atomic. There are no repeating groups or arrays.
- Additionally
    - All entries in a column are of the same data type.
    - Each column has a unique name.
    - The order in which data is stored does not affect the database's integrity.

  The model is in 1NF

2NF

- It is in 1NF.
- All non-key attributes are fully functionally dependent on the primary key.

  The model is in 2NF

3NF

- It is in 2NF.
- It has no transitive dependencies.

  The model is in 3NF

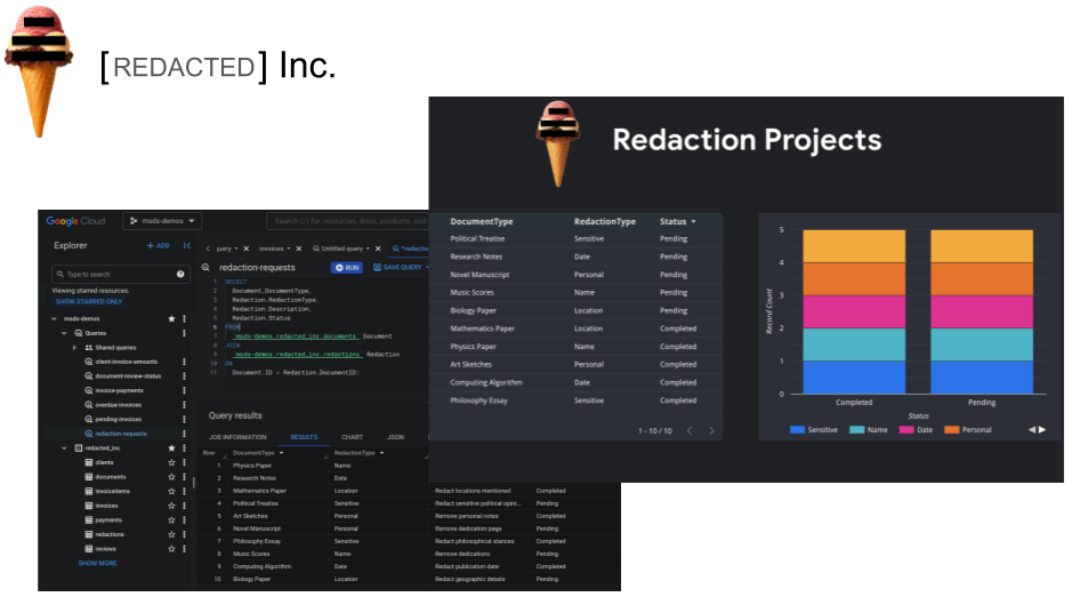
Step 6: Finalize the Relational Model in 3NF for Implementation

There was not updates to my initial RM needed for this step.

```
CLIENTS(ID (PK), Name, Email, Phone)
        FD: ID -> Name, Email, Phone

DOCUMENTS(ID (PK), ClientID (FK), DocumentType, SubmissionDate)
        FD: ID -> ClientID, DocumentType, SubmissionDate

REDACTIONS(ID (PK), DocumentID (FK), RedactionType, Description, Status)
        FD: ID -> DocumentID, RedactionType, Description, Status

REVIEWS(ID (PK), DocumentID (FK), ReviewDate, ApprovalStatus)
        FD: ID -> DocumentID, ReviewDate, ApprovalStatus

INVOICES(ID (PK), ClientID (FK), InvoiceDate, TotalAmount, DueDate, PaymentStatus)
        FD: ID -> ClientID, InvoiceDate, TotalAmount, DueDate, PaymentStatus

INVOICEITEMS(ID (PK), InvoiceID (FK), Description, Amount, Quantity)
        FD: ID -> InvoiceID, Description, Amount, Quantity

PAYMENTS(ID (PK), InvoiceID (FK), PaymentDate, Amount, PaymentMethod)
        FD: ID -> InvoiceID, PaymentDate, Amount, PaymentMethod
```

One Step Beyond

I further created mock data, implemented a BigQuery database, created business views (queries) and visualized a KPI in Looker Studio, based on the relational design for [REDATED] Inc to add some final color to the slides.



Thank you!

Thanks for learning a bit about [REDACTED] Inc. Good luck on your courses.