Basé sur l'image fournie qui montre le schéma d'une table (probablement Oracle via SQL*Plus ou un éditeur texte) nommée ID_CODE_TIERS_CRM, voici une analyse de **Data Profiling** axée sur les anomalies structurelles et les risques de qualité de données.

Comme je n'ai accès qu'à la structure (métadonnées) et non aux données elles-mêmes, voici les anomalies de conception et les problèmes potentiels que ce schéma suggère :

1. Anomalies de Normalisation et de Structure

- **Mélange d'Entités (Personnes vs Entreprises) :**
  - La colonne ID_ICE_CIN suggère que la table stocke à la fois des entreprises (identifiées par **ICE** - Identifiant Commun de l'Entreprise au Maroc) et des individus (identifiés par **CIN**).
  - *Risque :* Cela viole le principe d'atomicité. Si un tiers a un ICE et un CIN (ex: entrepreneur individuel), comment est-ce stocké ? De plus, il existe une autre colonne CIN plus bas. C'est une **redondance** potentielle.
- **Redondance des Identifiants :**
  - Il y a ID_CODE_TIERS_CRM (clé primaire probable) et TIERS_CODE. Avoir deux codes pour la même entité crée souvent des désalignements lors des jointures.
- **Préfixes Incohérents (Naming Convention) :**
  - Le début de la table suit une norme stricte : ID_ (Identité), CRD_ (Coordonnées), LOC_ (Localisation), GST_ (Gestion), QLT_ (Qualité).
  - Cependant, la fin de la table abandonne cette logique : PAYS, CAPITAL, SYNERGIE, REVENDEUR, SEGMENT n'ont pas de préfixe. Cela suggère que ces colonnes ont été ajoutées "à la va-vite" plus tard (Schema Drift).

2. Anomalies de Types de Données (Data Types)

- **Tailles de VARCHAR2 excessives ou arbitraires :**
  - ID_ICE_CIN est VARCHAR2(100). Or, un ICE fait 15 chiffres et une CIN moins de 10 caractères. Une taille de 100 permet de saisir du "bruit" (ex: "ICE: 123456...").
  - LOC_GPS est VARCHAR2(100). C'est très large pour des coordonnées. Cela suggère que les données ne sont pas normalisées (ex: "33.5731, -7.5898" vs "Lat: 33... Long: -7...").
- **Colonnes Booléennes Mal Définies :**
  - INTRAGROUPE est NUMBER(1). C'est correct pour un booléen (0/1), mais SYNERGIE, COMMUN, REVENDEUR sont en VARCHAR2(3).
  - *Anomalie :* Utiliser 3 caractères suggère qu'on stocke "OUI"/"NON" ou "YES"/"NO". C'est inefficace et sujet aux erreurs (ex: "OuI", "yes", "Y"). Il faut standardiser sur NUMBER(1) ou CHAR(1).
- **Précision Numérique Manquante :**
  - CAPITAL est défini comme NUMBER sans précision (ex: NUMBER(15,2)). Cela peut entraîner des notations scientifiques ou des erreurs d'arrondi sur des montants financiers.

3. Anomalies de Contraintes (Nullability)

- **Absence de NOT NULL critique :**
  - Seul ID_CODE_TIERS_CRM est marqué NOT NULL.
  - *Risque Majeur :* ID_RAISON_SOCIALE (Nom de l'entreprise) est nullable. Vous pouvez donc créer un client "fantôme" sans nom.
  - GST_DATE_CREATION est nullable. Il est anormal qu'un enregistrement n'ait pas de date de création (piste d'audit incomplète).

4. Anomalies Fonctionnelles (Business Logic)

- **Dédoublonnage flou :**
  - La présence de COUNT_MAPPING, ENTITE_MAPPEE, QLT_CODE_SOURCE suggère que cette table est le résultat d'une consolidation (Master Data Management).
  - Si ENTITE_MAPPEE est rempli, cela veut-il dire que cette ligne est un doublon ? La logique de "Golden Record" n'est pas claire dans le schéma.
- **Données Étrangères :**
  - IDENTIFIANT_ETRANGER et VILLE_ETRANGERE. Si le client est local, ces champs seront vides (Sparse Data). Il serait plus propre d'avoir une table satellite pour les adresses internationales si elles sont rares.

Plan d'action (Requêtes SQL de Profiling)

Pour confirmer ces hypothèses, voici les requêtes SQL que vous devriez lancer sur cette table :

1. **Vérifier les "Clients Fantômes" (Qualité critique) :**
   ```SQL
   SELECT count(*) FROM TABLE WHERE ID_RAISON_SOCIALE IS NULL;
   ```

2. **Vérifier la redondance ICE/CIN :**
   ```SQL
   SELECT ID_ICE_CIN, CIN, count(*)
   FROM TABLE
   WHERE ID_ICE_CIN IS NOT NULL AND CIN IS NOT NULL
   GROUP BY ID_ICE_CIN, CIN;
   ```

3. **Analyser la qualité des Coordonnées :**
   ```SQL
   -- Chercher les emails invalides (sans @)
   SELECT count(*) FROM TABLE WHERE CRD_EMAIL NOT LIKE '%@%';
   -- Chercher les numéros de téléphone trop courts
   SELECT count(*) FROM TABLE WHERE LENGTH(CRD_TELEPHONE) < 8;
   ```

4. **Vérifier la consistance des Booléens (VARCHAR2(3)) :**
   SQL
   ```
   SELECT DISTINCT SYNERGIE FROM TABLE;
   -- Vous trouverez surement : 'OUI', 'NON', 'Oui', NULL, 'N/A'
   ```

1. Le cas "Schizophrène" : Même ID, mais données différentes

Problème : Le système a inséré deux fois le même ID (ex: 1001), mais avec des informations potentiellement contradictoires (ex: une ligne dit "Client A", l'autre "Client B").

Requête pour isoler les ID dupliqués :

SQL

```
SELECT ID_CODE_TIERS_CRM, COUNT(*) as Nbr_Lignes
FROM VOTRE_TABLE
GROUP BY ID_CODE_TIERS_CRM
HAVING COUNT(*) > 1;
```

Requête pour voir les contradictions (Le même ID a-t-il des noms différents ?) :

SQL

```
SELECT
   A.ID_CODE_TIERS_CRM,
   A.ID_RAISON_SOCIALE as NOM_LIGNE_1,
   B.ID_RAISON_SOCIALE as NOM_LIGNE_2
FROM VOTRE_TABLE A
JOIN VOTRE_TABLE B ON A.ID_CODE_TIERS_CRM = B.ID_CODE_TIERS_CRM
WHERE A.ROWID < B.ROWID -- Pour éviter de comparer la ligne avec elle-même
AND A.ID_RAISON_SOCIALE <> B.ID_RAISON_SOCIALE; -- On cherche les différences
```

---

2. Le cas "Doublon Métier" : Deux IDs différents, même entité réelle

Problème : Le client "Maroc Telecom" existe sous l'ID 1001 et sous l'ID 5099. C'est un échec du MDM (Master Data Management).

Pour détecter cela, il faut ignorer l'`ID_CODE_TIERS_CRM` et chercher des doublons sur les **clés fonctionnelles** (ICE, CIN, Email, Téléphone).

A. Doublons par ICE / CIN (Le plus fiable pour le Maroc) :

SQL

```
SELECT ID_ICE_CIN, COUNT(DISTINCT ID_CODE_TIERS_CRM) as Nbr_IDs_Distincts
FROM VOTRE_TABLE
WHERE ID_ICE_CIN IS NOT NULL
AND LENGTH(ID_ICE_CIN) > 5 -- Pour éviter les faux doublons sur des valeurs courtes comme "0" ou "-"
GROUP BY ID_ICE_CIN
HAVING COUNT(DISTINCT ID_CODE_TIERS_CRM) > 1;
```

Interprétation : Si cette requête retourne des lignes, cela veut dire qu'une même entreprise a été créée plusieurs fois avec des IDs différents.

B. Doublons par Email ou Téléphone (Souvent source de création multiple) :

SQL

```
SELECT CRD_EMAIL, LISTAGG(ID_CODE_TIERS_CRM, ', ') WITHIN GROUP (ORDER BY ID_CODE_TIERS_CRM) as Liste_IDs
FROM VOTRE_TABLE
WHERE CRD_EMAIL IS NOT NULL AND CRD_EMAIL LIKE '%@%'
GROUP BY CRD_EMAIL
HAVING COUNT(DISTINCT ID_CODE_TIERS_CRM) > 1;
```

---

3. Analyse des "Faux NULLs" (Anomalies de saisie)

Souvent, pour contourner l'unicité, les utilisateurs saisissent des valeurs "poubelle". Il faut les identifier pour les nettoyer.

Requête pour trouver les données "bruitées" qui cachent des doublons :

```
SQL

SELECT ID_ICE_CIN, count(*)

FROM VOTRE_TABLE

WHERE ID_ICE_CIN IN ('.', '0', 'TEST', 'NC', 'N/A', 'INCONNU', '11111')

GROUP BY ID_ICE_CIN;
```

Résumé du Plan de Data Profiling

Si vous devez présenter un rapport d'anomalies, voici les indicateurs (KPIs) à calculer :

1. **Taux d'unicité ID :** (Nombre d'IDs uniques / Nombre total de lignes) * 100.
2. **Taux de duplication Client :** Nombre d'ICE associés à plus d'un ID Tiers.
3. **Volume de conflits :** Nombre d'IDs dupliqués ayant des `ID_RAISON_SOCIALE` différents.