Documentation du dataset hospitalier simulé

Période : 1er janvier 2022 → 31 décembre 2024

Objectif du dataset

Ce dataset a pour objectif de **simuler le fonctionnement quotidien d'un hôpital** sur plusieurs années, en intégrant :

- des règles calendaires
- des règles météo
- des règles sanitaires
- des contraintes de ressources humaines
- des contraintes physiques (lits)

Chaque jour est une observation indépendante mais **cohérente avec le jour précédent**.

📄 1) `daily_hospital_context`

Description générale

Feuille centrale du dataset.
👉 **1 ligne = 1 jour = 1 état global de l'hôpital**

Période couverte :

- `date ∈ [2022-01-01 ; 2024-12-31]`

🕑 Variables temporelles

`date`

- **Description** : Date calendaire observée.
- **Type** : date (`YYYY-MM-DD`)
- **Calcul** : Génération par itération quotidienne.
- **Règle** : dataset continu sans trous.

`jour_semaine`

- **Description** : Jour de la semaine correspondant à la date.
- **Type** : enum
- **Nomenclature** : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
- **Calcul** : Dérivé de `date`.

Règles associées

```
Si jour_semaine ∈ {Saturday, Sunday}
Alors :
  - effectifs disponibles ↓
  - admissions programmées ↓
  - passages urgences ↑ légèrement
```

`jour_mois`

- **Description** : Numéro du jour dans le mois.
- **Type** : int
- **Nomenclature** : 1–31
- **Calcul** : Dérivé de `date`.

`semaine_annee`

- **Description** : Numéro de semaine ISO.
- **Type** : int
- **Nomenclature** : 1–53
- **Calcul** : ISO week à partir de `date`.

`mois`

- **Description** : Numéro du mois.
- **Type** : int
- **Nomenclature** : 1–12
- **Calcul** : Dérivé de `date`.

`annee`

- **Description** : Année civile.
- **Type** : int
- **Valeurs possibles** : 2022, 2023, 2024
- **Calcul** : Dérivé de `date`.

`saison`

- **Description** : Saison météorologique.
- **Type** : enum

- **Nomenclature** : Hiver, Printemps, Été, Automne

Calcul (règle métier fixe)

```
Hiver      : 15 décembre → 15 mars
Printemps  : 16 mars → 15 juin
Été        : 16 juin → 15 septembre
Automne    : 16 septembre → 14 décembre
```

Règles associées

```
Si saison = Hiver
→ pathologies respiratoires ↑
→ passages urgences ↑

Si saison = Été
→ traumatologie ↑
→ déshydratation ↑
```

`vacances_scolaires`

- **Description** : Indique si la date est en période de vacances scolaires.
- **Type** : bool
- **Valeurs** : Oui / Non
- **Calcul** : Jointure avec `school_holidays_reference`.

Règles associées

```
Si vacances_scolaires = Oui
Alors :
  - nb_*_disponibles ↓
  - admissions programmées ↓
```

🌦️ Variables météo

`temperature_moyenne`

- **Description** : Température moyenne journalière en °C.
- **Type** : float
- **Calcul** :
    - soit donnée réelle (API météo)
    - soit tirage autour d'une moyenne saisonnière.

`temperature_min`

- **Description** : Température minimale journalière.
- **Type** : float
- **Calcul** :
    - `temperature_min = temperature_moyenne - δ`

`temperature_max`

- **Description** : Température maximale journalière.
- **Type** : float
- **Calcul** :
    - `temperature_max = temperature_moyenne + δ`

`meteo_principale`

- **Description** : Conditions météorologiques dominantes.
- **Type** : enum
- **Nomenclature** : Soleil, Pluie, Gris, Vent, Neige, Doux, Froid
- **Calcul** : Dépend de la saison + températures.

`indice_chaleur`

- **Description** : Indicateur normalisé de stress thermique chaud.
- **Type** : float `[0–1]`

Calcul

- `indice_chaleur = max(0, (temperature_max - 25) / 10)`

Règle

- `Si indice_chaleur > 0`
- `→ risque canicule`

`indice_froid`

- **Description** : Indicateur normalisé de stress thermique froid.
- **Type** : float `[0–1]`

Calcul

- `indice_froid = max(0, (0 - temperature_min) / 10)`

🏥 Capacités hospitalières

`lits_total`

- **Description** : Nombre total de lits disponibles.
- **Type** : int
- **Source** : `hospital_baseline`.

`lits_occupes`

- **Description** : Nombre de lits occupés ce jour-là.
- **Type** : int

Calcul

- `lits_occupes = lits_total × taux_occupation_lits`

`taux_occupation_lits`

- **Description** : Taux global d'occupation.
- **Type** : float `[0–1]`

Règles

- `Si saison = Hiver → taux ↑`
- `Si evenement_special actif → taux ↑`
- `Si taux > 0.95 → saturation`

👩‍⚕️ Ressources humaines

`nb_medecins_disponibles`

- **Description** : Nombre de médecins présents.
- **Type** : int

Calcul

- `nb_medecins_reference`
- `→ staff_variation_rules`
- `→ ajustement vacances / événements`

`nb_infirmiers_disponibles`

- **Description** : Nombre d'infirmiers présents.
- **Type** : int
- **Calcul** : Même logique que médecins.

`nb_aides_soignants_disponibles`

- **Description** : Nombre d'aides-soignants présents.
- **Type** : int
- **Calcul** : Même logique que médecins.

`taux_couverture_personnel`

- **Description** : Couverture RH par rapport au nominal.
- **Type** : float

Calcul

- `taux = personnel_disponible / personnel_reference`

Règle

- `Si taux < 0.85`
- `→ capacité d'admission ↓`

📊 Activité hospitalière

`nombre_admissions`

- **Description** : Admissions totales du jour.
- **Type** : int

Calcul

- `admissions_moyennes_jour`
- `× (1 + impact_evenement_estime)`
- `× bruit_aléatoire`

`nombre_passages_urgences`

- **Description** : Passages aux urgences.
- **Type** : int
- **Calcul** : Même logique que admissions avec coefficient spécifique.

`nombre_hospitalisations`

- **Description** : Admissions nécessitant un lit.

- **Type** : int

Calcul

  - `nombre_admissions × taux_hospitalisation`

Règle de cohérence

  - `nombre_hospitalisations ≤ lits_disponibles`

`nombre_sorties`

- **Description** : Sorties hospitalières.
- **Type** : int

Calcul

  - `nombre_sorties = nombre_hospitalisations / duree_sejour_moyenne`

🚨 Événements

`evenement_special`

- **Description** : Événement sanitaire ou environnemental actif.
- **Type** : enum
- **Nomenclature** :
  Aucun, Canicule, Epidemie_grippe, Epidemie_gastro, Pollen_Allergene, Vague_froid

Détection

  - `via event_detection_rules`

- `priorité la plus élevée si conflit`

`impact_evenement_estime`

- **Description** : Impact relatif sur les admissions.
- **Type** : float (ex : 0.20 = +20%)
- **Source** : `special_events_reference`.

PAGE 2 — `hospital_baseline`

Rôle de la page
Cette table définit la **configuration structurelle de référence de l'hôpital**.
Elle représente l'**état nominal** (normal) de l'établissement, indépendamment des variations journalières.

👉 Toutes les valeurs de cette table sont :

- **stables dans le temps** (2022–2024)
- utilisées comme **base de calcul** pour `daily_hospital_context`
- comparées aux valeurs journalières pour mesurer des écarts (sous-effectif, saturation, etc.)

🏥 Identification de l'établissement

`hospital_id`

- **Description** : Identifiant unique et technique de l'hôpital dans le dataset.
- **Type** : string
- **Nomenclature** : format libre mais unique (ex. `HOP_0001`)

- **Usage** :
    - clé primaire logique
    - clé de jointure avec `daily_hospital_context`

Règle :

`hospital_id est unique dans hospital_baseline`

-

`hospital_nom`

- **Description** : Nom complet et lisible de l'établissement hospitalier.
- **Type** : string
- **Usage** :
    - affichage
    - reporting
    - compréhension humaine
- **Source** : référentiel établissement.

`ville`

- **Description** : Ville d'implantation principale de l'hôpital.
- **Type** : string
- **Usage** :
    - contextualisation géographique
    - rattachement météo / vacances scolaires

`region`

- **Description** : Région administrative de rattachement.
- **Type** : string
- **Nomenclature** :
    - nom de région (ex. `Île-de-France`)
    - ou code INSEE / région SAE
- **Usage** :
    - agrégation territoriale
    - compatibilité SAE

`type_hopital`

- **Description** : Catégorie structurelle de l'établissement.
- **Type** : enum
- **Nomenclature** :
    - `CHU` (Centre Hospitalier Universitaire)
    - `CH` (Centre Hospitalier)
    - `Clinique`
    - `ESPIC`
    - `Autre`
- **Usage** :
    - différenciation des comportements (capacité, personnel, complexité)

Règle :

```
type_hopital influence indirectement
admissions_moyennes_jour
et nb_*_reference
```

-

🛏️ Capacités structurelles en lits

`lits_total_mco`

- **Description** : Nombre de lits dédiés à l'activité **Médecine, Chirurgie et Obstétrique**.
- **Type** : int
- **Unité** : nombre de lits
- **Source** : SAE ou paramétrage fixe.
- **Usage** :
    - calcul de la capacité d'hospitalisation aiguë
    - base de `lits_total`

Règle :

```
lits_total_mco > 0
```

-

`lits_urgences`

- **Description** : Nombre de lits ou places utilisés pour l'accueil et l'observation de courte durée aux urgences (incluant UHCD).
- **Type** : int

- **Unité** : lits / places
- **Source** : SAE.
- **Usage** :
    - contextualisation des passages urgences

Règle :

```
lits_urgences ≤ lits_total_mco
```

-

---

lits_reanimation

- **Description** : Nombre de lits de réanimation (soins critiques).
- **Type** : int
- **Source** : SAE.
- **Usage** :
    - indicateur de capacité critique
    - utilisé implicitement lors de pics (épidémies, vagues de froid)

Règle :

```
lits_reanimation << lits_total_mco
```

-

---

lits_ssr

- **Description** : Nombre de lits de soins de suite et de réadaptation.
- **Type** : int
- **Source** : SAE.

- **Usage** :
    - calcul des sorties
    - durée de séjour moyenne

Règle :

```
lits_ssr ≥ 0
```

-

👩‍⚕️ Effectifs de référence

> Ces effectifs représentent la **dotation normale**, hors absences, congés, événements.

`nb_medecins_reference`

- **Description** : Nombre moyen de médecins constituant l'effectif nominal.
- **Type** : int
- **Source** : SAE (Q21–Q24 / PCAMEDURG) ou estimation.
- **Usage** :
    - base de calcul pour `nb_medecins_disponibles`

Règle :

```
nb_medecins_reference > 0
```

-

`nb_infirmiers_reference`

- **Description** : Nombre moyen d'infirmiers constituant l'effectif nominal.

- **Type** : int
- **Source** : SAE ou estimation.
- **Usage** :
    - base RH principale pour la capacité de prise en charge

Règle :

```
nb_infirmiers_reference > nb_medecins_reference
```

-

`nb_aides_soignants_reference`

- **Description** : Nombre moyen d'aides-soignants constituant l'effectif nominal.
- **Type** : int
- **Source** : SAE ou estimation.
- **Usage** :
    - soutien opérationnel aux soins

Règle :

```
nb_aides_soignants_reference ≥ nb_infirmiers_reference
```

-

📊 Activité moyenne de référence

`admissions_moyennes_jour`

- **Description** : Nombre moyen d'admissions hospitalières par jour en situation normale.
- **Type** : float

Calcul :

```
admissions_annuelles / 365 (ou 366)
```

-
- **Usage** :
    - point d'ancrage du volume journalier

Règle :

```
admissions_moyennes_jour > 0
```

-

`passages_urgences_moyens_jour`

- **Description** : Nombre moyen de passages aux urgences par jour.
- **Type** : float

Calcul :

```
passages_urgences_annuels / 365 (ou 366)
```

-
- **Usage** :
    - base de `nombre_passages_urgences`

`duree_sejour_moyenne`

- **Description** : Durée moyenne de séjour d'un patient hospitalisé.
- **Type** : float
- **Unité** : jours
- **Source** : estimation ou paramétrage.

- **Usage** :
    - calcul de `nombre_sorties`

Règle :

```
duree_sejour_moyenne > 0
```

-

🔗 Relations avec les autres pages

- Utilisée par :
    - `daily_hospital_context`
    - `staff_variation_rules`
- Fournit :
    - capacités maximales
    - valeurs de référence
- Ne dépend d'aucune autre table.

PAGE 3 — `staff_variation_rules`

Rôle de la page
Cette table définit **comment varier les effectifs disponibles** (médecins / infirmiers / aides-soignants) selon :

- une **période** (mois, saison)
- ou un **événement** (grippe, canicule, etc.)

👉 Elle sert à générer, pour chaque jour (2022–2024), les variables :

- `nb_medecins_disponibles`

- `nb_infirmiers_disponibles`
- `nb_aides_soignants_disponibles`

dans `daily_hospital_context`.

Concept : distribution par tranche

Chaque règle décrit **une tranche d'effectif** :

- bornée par `borne_min` et `borne_max`
- avec une `probabilite`

💡 Exemple (juillet, infirmiers) :

- 80% de chance d'être dans [230–270]
- 10% dans [200–229]
- 10% dans [271–290]

Puis le générateur :

1. tire une tranche selon la probabilité
2. tire une valeur dans la tranche (uniforme ou normal tronquée)

Variables

`type_personnel`

- **Description** : Catégorie de personnel concernée par la règle.
- **Type** : enum
- **Nomenclature** :

    - `medecin`
    - `infirmier`
    - `aide_soignant`
- **Usage** :
    - filtre la règle à appliquer selon le type de personnel à générer

Règle de cohérence :

```
type_personnel ∈ {medecin, infirmier, aide_soignant}
```

-

`periode`

- **Description** : Contexte temporel ou événementiel sur lequel s'applique la règle.
- **Type** : string
- **Nomenclature (formats autorisés)** :
    - `mois=01` à `mois=12`
    - `saison=Hiver|Printemps|Été|Automne`
    - `evenement=Canicule|Epidemie_grippe|Epidemie_gastro|Pollen_Allergene|Vague_froid|Autre`
- **Usage** :
    - sélection de la distribution correcte pour le jour courant

Règle de cohérence :

```
periode respecte un des formats :

mois=XX | saison=YYY | evenement=ZZZ
```

-

Règle associée (priorité de sélection)
Quand plusieurs périodes pourraient s'appliquer (ex : mois + saison + événement), on applique une priorité :

```
evenement=...  >  mois=...  >  saison=...
```

(Ex : si Canicule active, elle override la distribution mensuelle.)

---

borne_min

- **Description** : Valeur minimale possible de l'effectif pour cette tranche.
- **Type** : int
- **Usage** :
    - encadre le tirage d'effectif

Règles :

```
borne_min ≥ 0
```

```
borne_min ≤ borne_max
```

-

---

borne_max

- **Description** : Valeur maximale possible de l'effectif pour cette tranche.
- **Type** : int
- **Usage** :
    - encadre le tirage d'effectif

Règles :

```
borne_max ≥ 0
```

`borne_max ≥ borne_min`

-

`probabilite`

- **Description** : Probabilité que l'effectif réel tombe dans la tranche [borne_min, borne_max].
- **Type** : float
- **Domaine** : `[0–1]`
- **Usage** :
    2. pondération lors du choix de tranche
- **Règles** :
    2. bornage

`0 ≤ probabilite ≤ 1`

-
    2. normalisation

`Pour un même couple (type_personnel, periode),`

`Σ probabilite = 1`

-

Calcul d'application (moteur)

Étape 1 — Identifier la période active pour la journée J

Pour une date `date_J` dans 2022–2024, on détermine :

- `mois=MM` via `date_J`

- `saison=...` via la règle saison
- `evenement=...` via `event_detection_rules` (si actif)

Étape 2 — Sélectionner les règles applicables

On cherche les règles dans cet ordre :

1. règles `evenement=...` si un événement est actif
2. sinon règles `mois=MM`
3. sinon règles `saison=...`
4. sinon fallback (ex : distribution globale “default”)

Étape 3 — Tirer une tranche

Soit l’ensemble des lignes trouvées :

```
[(borne_min_i, borne_max_i, probabilite_i) pour i=1..n]
```

On tire un index i selon `probabilite_i`.

Étape 4 — Tirer un effectif dans la tranche

Deux méthodes acceptables (à choisir dans `generation_parameters`) :

Option A — Uniforme

```
effectif = randint(borne_min, borne_max)
```

Option B — Normal tronquée (plus réaliste)

- centre sur la moyenne de la tranche

- limite aux bornes

```
mu = (borne_min + borne_max)/2

sigma = (borne_max - borne_min)/6    # approx

effectif = clip(round(N(mu, sigma)), borne_min, borne_max)
```

Étape 5 — Ajustements contextuels (post-tirage)

Après tirage, on applique des modificateurs journaliers :

Week-end

```
Si jour_semaine ∈ {Saturday, Sunday}

→ effectif *= (0.85 à 0.95)
```

Vacances scolaires

```
Si vacances_scolaires = Oui

→ effectif *= (0.80 à 0.95)
```

Neige / conditions difficiles

```
Si meteo_principale ∈ {Neige}

→ effectif *= (0.80 à 0.95)
```

(les coefficients exacts peuvent être stockés dans `generation_parameters`)

Règles métier “exemple soignants” (ton cas)

Tu avais :

- historique : 20, 18, 12, 25 en juillet selon années
- règle : 80% [18–22], 10% <18, 10% >22

Traduction en lignes `staff_variation_rules` (exemple) :

- `periode = mois=07`
- `type_personnel = aide_soignant` (ou infirmier selon ce que tu veux modéliser)
- tranches :
  - `[18,22] prob=0.80`
  - `[12,17] prob=0.10`
  - `[23,25] prob=0.10`

Contrôles qualité (recommandés)

À faire dans le pipeline :

Complétude :

```
pour chaque type_personnel et chaque mois,
au moins 1 règle existe
```

-

Normalisation :

```
Σ prob = 1 à 1e-6 près
```

-

Bornes :

```
borne_min ≤ borne_max
```

-

🔗 Relations avec les autres pages

- Dépend de :
    - `daily_hospital_context` (calendrier : mois/saison/week-end)
    - `event_detection_rules` (événements actifs)
    - `generation_parameters` (choix uniforme/normal, coefficients ajustements)
- Alimente :
    - `daily_hospital_context.nb_*_disponibles`

PAGE 4 — `weather_daily_reference`

Rôle de la page
Cette table contient les **données météorologiques journalières** utilisées comme **entrée exogène** du système.
Elle sert à :

- caractériser le contexte environnemental d'un jour
- déclencher des **événements climatiques** (canicule, vague de froid, neige…)
- influencer indirectement :
    - les admissions
    - la disponibilité du personnel
    - la pression hospitalière

👉 Une ligne = **une journée météo**
👉 Période couverte : **01/01/2022 → 31/12/2024**

🌦️ Principes généraux

- La météo est considérée comme **indépendante de l'hôpital**
- Elle peut être :
    - issue d'une **API météo réelle**
    - ou **simulée** à partir de règles saisonnières
- Elle ne dépend d'aucune autre table
- Elle est **jointe par la date** à `daily_hospital_context`

Variables

`date`

- **Description** : Date calendaire correspondant à l'observation météo.
- **Type** : date (`YYYY-MM-DD`)
- **Calcul** : Génération ou import sur toute la période 2022–2024.

Règle :

```
date unique dans weather_daily_reference
```

-

`temperature_moyenne`

- **Description** : Température moyenne journalière en degrés Celsius.
- **Type** : float
- **Unité** : °C
- **Origine possible** :
    - donnée réelle (API météo)

- ou simulation

Simulation (si pas d'API)

```
temperature_moyenne = moyenne_saisonniere
                      + bruit_gaussien(σ dépendant de la saison)
```

Exemple de moyennes saisonnières (indicatif) :

- Hiver : 5°C
- Printemps : 12°C
- Été : 22°C
- Automne : 14°C

`temperature_min`

- **Description** : Température minimale observée sur la journée.
- **Type** : float
- **Unité** : °C

Calcul

```
temperature_min = temperature_moyenne - delta_min
```

où `delta_min` est tiré aléatoirement (ex : 2–7°C).

Règle de cohérence

```
temperature_min ≤ temperature_moyenne
```

`temperature_max`

- **Description** : Température maximale observée sur la journée.
- **Type** : float
- **Unité** : °C

Calcul

```
temperature_max = temperature_moyenne + delta_max
```

où `delta_max` est tiré aléatoirement (ex : 2–8°C).

Règle de cohérence

```
temperature_max ≥ temperature_moyenne
```

`meteo_principale`

- **Description** : Type de conditions météorologiques dominantes sur la journée.
- **Type** : enum
- **Nomenclature** :
    - Soleil
    - Nuage
    - Gris
    - Pluie
    - Vent
    - Neige
    - Orage

Calcul (logique simplifiée)

```
Si temperature_max ≥ 20 et précipitations = 0

→ Soleil


Si précipitations > 0

→ Pluie ou Orage


Si temperature_min ≤ 0

→ Neige


Sinon

→ Nuage ou Gris
```

Usage

- détection canicule
- détection vague de froid
- ajustement staff (conditions difficiles)

`vent`

- **Description** : Vitesse moyenne du vent sur la journée.
- **Type** : float
- **Unité** : km/h

Calcul

```
vent = tirage aléatoire selon saison
```

Exemples :

- Été : 5–25 km/h
- Hiver : 10–50 km/h

Règle

```
vent élevé + froid → conditions difficiles
```

---

humidite

- **Description** : Taux d'humidité moyen journalier.
- **Type** : int
- **Unité** : %
- **Domaine** : 0–100

Calcul

```
humidite = tirage conditionné par meteo_principale
```

Exemples :

- Pluie : 70–100%
- Soleil : 30–60%

---

pression

- **Description** : Pression atmosphérique moyenne journalière.
- **Type** : int

- **Unité** : hPa

Calcul

```
pression = tirage autour de 1013 hPa
```

Usage

- purement contextuel
- non critique pour les règles métier actuelles
- conservé pour extensibilité future

🔥 Règles climatiques dérivées (utilisées ailleurs)

Les variables de cette table alimentent directement :

Détection canicule

```
Si temperature_max > 25°C pendant ≥ 1 jour

ET meteo_principale = Soleil

→ Canicule possible


OU


Si temperature_moyenne > 23.9°C pendant ≥ 3 jours

ET meteo_principale = Soleil

→ Canicule confirmée
```

Détection vague de froid

```
Si temperature_min < 0°C pendant ≥ 2 jours

→ Vague_froid
```

Impact indirect sur le personnel

```
Si meteo_principale ∈ {Neige, Orage violent}

→ nb_*_disponibles ↓
```

🔗 Relations avec les autres pages

- **Utilisée par** :
  - `daily_hospital_context`
  - `event_detection_rules`
- **Indépendante de** :
  - hospital_baseline
  - staff_variation_rules
- **Clé de jointure** :

  `date`

PAGE 5 — `school_holidays_reference`

Rôle de la page
Cette table contient le **calendrier de référence des vacances scolaires**.

Elle sert à déterminer, pour chaque jour entre **2022-01-01 et 2024-12-31**, si l'on est en période de vacances, et quel type de vacances s'applique.

👉 Impact principal dans le modèle :

- baisse de disponibilité RH (congés)
- baisse d'activité programmée
- légère hausse possible de passages urgences certains jours (effet indirect, optionnel)

Principes généraux

- C'est une **table "vérité terrain"** : pas de calcul à partir d'autres variables.
- Les données peuvent être :
  - importées d'un calendrier officiel
  - saisies manuellement
- On joint cette table à `daily_hospital_context` via :
  - `date`
  - et éventuellement `zone` (si multi-zones)

👉 Si ton dataset est centré sur Paris, tu peux fixer la zone à **Zone C** (Île-de-France), mais garder la colonne `zone` pour extensibilité.

Variables

`date`

- **Description** : Date calendaire pour laquelle on indique le statut vacances.
- **Type** : date (`YYYY-MM-DD`)

Règle :

```
1 ligne par (date, zone)

date ∈ [2022-01-01 ; 2024-12-31]
```

-

---

zone

- **Description** : Zone académique à laquelle s'applique le calendrier.
- **Type** : enum
- **Nomenclature** :
    - Zone A
    - Zone B
    - Zone C
- **Usage** :
    - permet d'adapter à différentes régions françaises
    - permet la jointure correcte si plusieurs hôpitaux de zones différentes

Règle de cohérence

```
zone ∈ {Zone A, Zone B, Zone C}
```

---

vacances_scolaires

- **Description** : Indique si la date tombe dans une période de vacances scolaires.
- **Type** : bool
- **Valeurs** : Oui / Non

- **Calcul** : valeur issue du calendrier (pas calculée).
- **Usage** :
    - injectée dans `daily_hospital_context.vacances_scolaires`

Règle associée (effet RH & activité)

```
Si vacances_scolaires = Oui

Alors :

  - nb_*_disponibles ↓ (ex : -5% à -20%)

  - admissions programmées ↓
```

`type_vacances`

- **Description** : Type de vacances scolaires correspondant à la date (si vacances_scolaires = Oui).
- **Type** : enum
- **Nomenclature** :
    - Toussaint
    - Noel
    - Hiver
    - Printemps
    - Été
- **Calcul** : valeur issue du calendrier (pas calculée).

Règles de cohérence

1. si pas vacances, type doit être vide ou “Aucun”

```
Si vacances_scolaires = Non

→ type_vacances = "Aucun" (recommandé) ou NULL
```

2. si vacances, type doit être renseigné

```
Si vacances_scolaires = Oui

→ type_vacances ∈ {Toussaint, Noel, Hiver, Printemps, Été}
```

Règles métier possibles (optionnel)
Tu peux introduire un effet différent selon le type :

```
Si type_vacances = Été

→ baisse RH plus forte (congés d'été)


Si type_vacances = Noel

→ baisse RH + activité programmée ↓ + urgences légèrement ↑ (jours fériés)
```

Calcul / utilisation dans `daily_hospital_context`

Jointure

Pour une date `d` et une zone `z` :

```
daily_hospital_context(d, z) LEFT JOIN school_holidays_reference(d, z)
```

Variables produites

- `vacances_scolaires` injectée telle quelle
- éventuellement `type_vacances` si tu veux l'exposer (sinon interne)

Contrôles qualité recommandés

- **Couverture temporelle**

```
Pour chaque zone utilisée :
  toutes les dates 2022-2024 doivent exister
```

- **Unicité**

```
(date, zone) est unique
```

- **Cohérence logique**

```
vacances_scolaires = Non → type_vacances = "Aucun" ou NULL
```

🔗 Relations avec les autres pages

- **Utilisée par**
  - `daily_hospital_context` (vacances_scolaires)
  - `staff_variation_rules` (ajustement RH post-tirage)
- **Ne dépend de rien**
- **Clés de jointure**

    - `date`
    - `zone` (si multi-zones)

PAGE 6 — `special_events_reference`

> **Rôle de la page**
> Cette table définit les **événements sanitaires et environnementaux exceptionnels** susceptibles d'avoir un **impact significatif sur l'activité hospitalière**.

👉 Elle ne sert **pas à détecter** les événements (ça, c'est la page 7),
👉 Elle sert à **décrire et paramétrer leurs effets** une fois qu'ils sont actifs.

C'est une table **semi-statique**, liée à la période **2022–2024**, utilisée comme :

- référentiel métier
- source d'impact chiffré sur les admissions
- documentation des hypothèses

Principes généraux

- Un événement :
    - a un **type**
    - peut être **daté explicitement** (si connu)
    - ou déclenché dynamiquement (via page 7)
- L'impact est exprimé **en variation relative** des admissions
- L'impact réel est **tiré aléatoirement dans une plage**

Variables

`evenement_type`

- **Description** : Type d'événement sanitaire ou environnemental.
- **Type** : enum
- **Nomenclature** :
    - Canicule
    - Epidemie_grippe
    - Epidemie_gastro
    - Pollen_Allergene
    - Vague_froid
    - Autre
- **Usage** :
    - clé logique de l'événement
    - jointure avec `event_detection_rules`

Règle de cohérence :

`evenement_type unique par ligne`

-

`date_debut`

- **Description** : Date de début de validité de l'événement (si connue).
- **Type** : date
- **Usage** :
    - permet de forcer un événement historique
    - utile pour des scénarios contrôlés

Règle :

```
date_debut ≤ date_fin
```

-

`date_fin`

- **Description** : Date de fin de validité de l'événement (si connue).
- **Type** : date
- **Usage** :
    - borne temporelle d'application de l'impact

Règle :

```
date_fin ≥ date_debut
```

-

💡 **Note dev**
Si `date_debut` et `date_fin` sont NULL →
l'événement est **100% piloté par les règles de détection** (page 7).

`condition_declenchement`

- **Description** : Description textuelle (humaine) des conditions de déclenchement.
- **Type** : string
- **Usage** :
    - documentation
    - traçabilité des hypothèses

Important :

```
cette colonne n'est PAS utilisée par le moteur
```

-

Exemples :

- “Températures élevées persistantes en été”
- “Pic saisonnier de pathologies respiratoires”

Impact sur l’activité

`impact_admissions_min`

- **Description** : Impact minimal estimé de l’événement sur les admissions.
- **Type** : float
- **Unité** : proportion
- **Exemple** :
    - `0.10` = +10 %

Règle

`impact_admissions_min ≥ 0`

`impact_admissions_max`

- **Description** : Impact maximal estimé de l’événement sur les admissions.
- **Type** : float
- **Unité** : proportion
- **Exemple** :
    - `0.30` = +30 %

Règles

`impact_admissions_max ≥ impact_admissions_min`

Calcul de l'impact effectif

Lorsqu'un événement est actif un jour `J` :

```
impact_evenement_estime_J
  = tirage aléatoire dans
    [impact_admissions_min ; impact_admissions_max]
```

Méthodes possibles :

- uniforme
- normale tronquée (préférée pour stabilité)

Exemple :

```
Canicule : min=0.15, max=0.30
→ impact tiré = 0.22 (+22%)
```

Exemples d'événements (recommandés)

Canicule

- `evenement_type` : Canicule
- `impact_admissions_min` : 0.15
- `impact_admissions_max` : 0.30

- **Effets indirects** :
    - admissions ↑
    - personnel ↓ (fatigue, congés)
    - durée séjour ↑ possible

Epidemie_grippe

- `evenement_type` : Epidemie_grippe
- `impact_admissions_min` : 0.20
- `impact_admissions_max` : 0.40
- **Saisonnalité** : hiver
- **Effets indirects** :
    - passages urgences ↑
    - lits MCO saturés

Epidemie_gastro

- `evenement_type` : Epidemie_gastro
- `impact_admissions_min` : 0.10
- `impact_admissions_max` : 0.25
- **Saisonnalité** : hiver / inter-saison
- **Effets indirects** :
    - urgences pédiatriques ↑

Pollen_Allergene

- `evenement_type` : Pollen_Allergene
- `impact_admissions_min` : 0.05
- `impact_admissions_max` : 0.15
- **Saisonnalité** : printemps
- **Effets indirects** :
    - urgences respiratoires ↑
    - effet modéré mais étalé

Vague_froid

- `evenement_type` : Vague_froid
- `impact_admissions_min` : 0.10
- `impact_admissions_max` : 0.25
- **Effets indirects** :
    - traumatologie (chutes) ↑
    - pathologies respiratoires ↑

Règles métier globales

Un seul événement actif à la fois

`Si plusieurs événements sont détectés le même jour`

`→ on conserve celui avec la priorité la plus élevée`

(la priorité est définie dans `event_detection_rules`)

🔗 Relations avec les autres pages

- **Utilisée par**
  - `daily_hospital_context` (impact_evenement_estime)
- **Liée à**
  - `event_detection_rules` (type d'événement)
- **Indépendante de**
  - hospital_baseline
  - staff_variation_rules
- **Clé logique**
  - `evenement_type`

PAGE 7 — `event_detection_rules`

> **Rôle de la page**
> Cette table définit **les règles logiques** permettant de déterminer, pour chaque jour `date` (2022–2024), si un **événement spécial** est actif, à partir du contexte météo et calendaire.

👉 Elle sert à calculer dans `daily_hospital_context` :

- `evenement_special`
- et donc à activer `impact_evenement_estime` (via `special_events_reference`)

Principes généraux

1) Détection = logique (IF/THEN)

Chaque règle décrit un événement et des conditions :

- température
- durée
- météo dominante

2) Gestion des conflits (priorité)

Plusieurs événements peuvent théoriquement être vrais le même jour.
On résout avec `priorite_evenement` :

`l'événement retenu = celui avec la priorité la plus élevée`

3) Source des variables d'entrée

Les conditions se basent principalement sur :

- `weather_daily_reference` (températures, meteo_principale)
- `daily_hospital_context` (saison, vacances_scolaires, jour_semaine)
- éventuellement une mémoire glissante (ex : "sur 3 jours") via séries temporelles

---

Variables

`evenement_type`

- **Description** : Type d'événement que la règle cherche à détecter.
- **Type** : enum
- **Nomenclature** :
    - Canicule
    - Epidemie_grippe

    - Epidemie_gastro
    - Pollen_Allergene
    - Vague_froid
    - Autre
- **Usage** :
    - identifie la sortie possible dans `daily_hospital_context.evenement_special`
    - permet de récupérer l'impact dans `special_events_reference`

Règle :

`evenement_type ∈ nomenclature`

-

`condition_temperature`

- **Description** : Condition sur les températures (min, max, moyenne) utilisée pour détecter l'événement.
- **Type** : string
- **Format** : expression lisible (style pseudo-code)
- **Exemples** :
    - `temperature_max > 25`
    - `temperature_min < 0`
    - `temperature_moyenne > 23.9`
- **Usage** :
    - le moteur peut parser cette condition (optionnel)

- sinon elle sert de documentation et la logique est codée en dur

---

`condition_duree`

- **Description** : Condition de durée (nombre de jours consécutifs) nécessaire au déclenchement.
- **Type** : string
- **Exemples** :
  - `>= 1 jour`
  - `>= 2 jours`
  - `>= 3 jours`
- **Usage** :
  - nécessite une fenêtre glissante sur la série météo

`condition_meteo`

- **Description** : Condition sur la météo dominante.
- **Type** : string
- **Exemples** :
  - `meteo_principale == "Soleil"`
  - `meteo_principale IN {"Pluie","Gris"}`
- **Usage** :
  - renforce la précision (ex : canicule = chaud + soleil)

---

priorite_evenement

- **Description** : Niveau de priorité pour arbitrer en cas de conflit.
- **Type** : int
- **Convention** : plus le chiffre est grand, plus la priorité est forte.
- **Usage** :
    - sélection de l'événement final journalier

Exemple de priorité recommandée

- 100 : Canicule
- 90 : Epidemie_grippe
- 80 : Vague_froid
- 70 : Epidemie_gastro
- 60 : Pollen_Allergene
- 10 : Autre

---

evenement_declenche

- **Description** : Booléen indiquant si la règle déclenche effectivement l'événement.
- **Type** : bool
- **Usage** :
    - plutôt utilisé pour debug / audit
    - en production, on le recalcule chaque jour au lieu de le stocker

Règle

`evenement_declenche n'est pas une donnée source,`

`c'est une sortie temporaire de moteur (optionnelle).`

Règles métier à implémenter (celles du projet + nouvelles)

> Les règles ci-dessous sont la traduction directe de ce que vous avez défini.

✅ Règle 1 — Canicule

Définition (celle que vous avez)

```
Si temperature_max > 25.0 pendant ≥ 1 jour

ET meteo_principale == "Soleil"

Alors Canicule


OU


Si temperature_moyenne > 23.9 pendant ≥ 3 jours

ET meteo_principale == "Soleil"

Alors Canicule
```

Notes dev

- nécessite une fenêtre glissante sur 3 jours
- peut être codée via :
    - rolling window sur temperature_moyenne
    - condition sur meteo_principale majoritaire

✅ Règle 2 — Vague de froid (nouvelle règle)

```
Si temperature_min < 0°C pendant ≥ 2 jours

Alors Vague_froid
```

Effets attendus

- admissions ↑
- passages urgences ↑
- traumatologie ↑ en cas de neige/verglas (optionnel)

✅ Règle 3 — Neige / verglas (règle “contexte”, optionnel)

Tu peux la modéliser soit comme un événement autonome “Autre”, soit comme un modificateur.

Option A (événement “Autre”)

```
Si meteo_principale == "Neige"

Alors Autre (type = Neige/Verglas)
```

Option B (modificateur d’activité sans événement)

```
Si meteo_principale == "Neige"

Alors passages urgences ↑

Et personnel disponible ↓
```

✅ Règle 4 — Epidemie de grippe (nouvelle règle)

Comme on n’a pas de vraie donnée épidémio, on utilise une règle “proxy” saisonnière réaliste :

```
Si saison == "Hiver"

ET temperature_moyenne < 10

ET meteo_principale IN {"Gris","Pluie","Nuage"}

Alors Epidemie_grippe (probabilité d'activation journalière)
```

Version probabiliste (recommandée)

```
Si conditions vraies

→ 30% de chance d'activer la grippe

Si activée, dure 7 à 21 jours (tirage)
```

(Ça rend l'épisode cohérent dans le temps au lieu de "on/off" chaotique.)

✅ Règle 5 — Epidemie de gastro (nouvelle règle)

```
Si saison IN {"Hiver","Printemps"}

ET temperature_moyenne < 15

ET humidite > 70

Alors Epidemie_gastro (probabilité d'activation)
```

Version probabiliste recommandée

- activation 20% si conditions réunies
- durée 5 à 14 jours

✅ Règle 6 — Pollen / Allergènes (nouvelle règle)

```
Si saison == "Printemps"
```

```
ET temperature_moyenne > 15

ET meteo_principale IN {"Soleil","Doux"}

Alors Pollen_Allergene (probabilité d'activation)
```

Version probabiliste recommandée

- activation 25%
- durée 10 à 30 jours (période étalée)

Gestion des conflits (priorités)

Si plusieurs événements sont détectés le même jour :

```
evenement_special = argmax(priorite_evenement)
```

Règle recommandée

```
Canicule > Epidemie_grippe > Vague_froid > Epidemie_gastro >
Pollen_Allergene > Autre
```

Sortie du moteur (daily_hospital_context)

Pour chaque date `d` :

1. on calcule quels événements sont vrais (bool)
2. on applique la règle de priorité
3. on retourne un seul `evenement_special`
4. on récupère `impact_evenement_estime` via `special_events_reference`

Contrôles qualité recommandés

- **Pas d'événement impossible**

```
Canicule en hiver : autorisé mais rare (si météo le permet)

Vague_froid en été : autorisé mais rare
```

- **Stabilité temporelle (anti-bruit)**
  Pour éviter l'activation/désactivation chaotique :
- événements doivent durer un minimum (ex : 3 jours)
- utiliser une logique "épisode" : start → durée → end

🔗 Relations avec les autres pages

- **Dépend de**
  - `weather_daily_reference` (températures, météo, humidité)
  - `daily_hospital_context` (saison, calendrier)
- **Alimente**
  - `daily_hospital_context.evenement_special`
- **Connecté à**
  - `special_events_reference` (impact min/max)
- **Utilise**
  - `priorite_evenement` pour arbitrage

PAGE 8 — `generation_parameters`

Rôle de la page
Cette table stocke les **paramètres globaux** utilisés pour générer le dataset de manière :

- **reproductible** (seed)
- **configurable** (mode de génération)
- **traçable** (documentation des hypothèses)

👉 L'objectif est que **le générateur n'ait pas de constantes "cachées" dans le code** : tout ce qui influence la simulation doit pouvoir être loggé ici.

Principes généraux

- Table **clé/valeur**
- Chaque paramètre a :
  - un nom (`parametre`)
  - une valeur (`valeur`) stockée en string (puis cast)
  - une description (`description`) pour l'usage humain
- Période couverte : **01/01/2022 → 31/12/2024**
- Utilisée par :
  - génération météo (si simulée)
  - génération staff (uniforme vs normal, coefficients)
  - bruit admissions
  - taux hospitalisation
  - règles de saturation / lissage

Variables

`parametre`

- **Description** : Nom unique d'un paramètre global.
- **Type** : string

Règle :

```
parametre est unique dans generation_parameters
```

-
- **Usage** : clé de configuration lue par le générateur.

`valeur`

- **Description** : Valeur associée au paramètre (stockée en texte puis convertie).
- **Type** : string
- **Usage** :
    - cast en int/float/bool/enum selon `parametre`

Règle :

```
la conversion doit être déterministe et validée
(sinon erreur de génération)
```

-

`description`

- **Description** : Explication lisible du rôle du paramètre.
- **Type** : string

- **Usage** : documentation humaine / audit / rapport.

Paramètres recommandés (MVP + réalismes)

Périmètre temporel

`date_debut`

- **Type attendu** : date string `YYYY-MM-DD`
- **Valeur recommandée** : `2022-01-01`
- **Description** : Date de début de la simulation.

`date_fin`

- **Type attendu** : date string `YYYY-MM-DD`
- **Valeur recommandée** : `2024-12-31`
- **Description** : Date de fin de la simulation.

Reproductibilité

`seed_random`

- **Type attendu** : int
- **Description** : Graine aléatoire pour reproduire exactement le même dataset.

Règle

```
Si seed_random fixé → génération déterministe

Si seed_random NULL → génération non reproductible
```

Mode de génération

`mode_generation`

- **Type attendu** : enum
- **Valeurs possibles** :
    - `api_meteo` (météo réelle)
    - `simulation_meteo` (météo synthétique)
    - `hybride` (réel partiel + simulation)
- **Description** : Indique la stratégie générale de génération.

Bruit sur admissions

`taux_bruit_admissions`

- **Type attendu** : float
- **Exemple** : `0.15`
- **Description** : Amplitude relative du bruit journalier sur les admissions.

Usage (exemple)

```
bruit_aléatoire ~ Normal(0, taux_bruit_admissions) tronqué

nombre_admissions *= (1 + bruit_aléatoire)
```

Lissage temporel (anti-sauts irréalistes)

`lissage_max_variation_jour`

- **Type attendu** : float

- **Exemple** : `0.40`
- **Description** : Limite max de variation relative entre J-1 et J.

Règle

`|admissions(J) - admissions(J-1)| / admissions(J-1) ≤ lissage_max_variation_jour`

Paramètres RH (post-tirage)

`coef_weekend_personnel_min`

- **Type attendu** : float
- **Exemple** : `0.85`
- **Description** : Coefficient minimum appliqué aux effectifs le week-end.

`coef_weekend_personnel_max`

- **Type attendu** : float
- **Exemple** : `0.95`
- **Description** : Coefficient maximum appliqué aux effectifs le week-end.

`coef_vacances_personnel_min`

- **Type attendu** : float
- **Exemple** : `0.80`
- **Description** : Coefficient minimum appliqué pendant les vacances scolaires.

`coef_vacances_personnel_max`

- **Type attendu** : float

- **Exemple** : `0.95`
- **Description** : Coefficient maximum appliqué pendant les vacances scolaires.

`coef_meteo_difficile_personnel_min`

- **Type attendu** : float
- **Exemple** : `0.80`
- **Description** : Coef minimum quand météo = neige / conditions difficiles.

`coef_meteo_difficile_personnel_max`

- **Type attendu** : float
- **Exemple** : `0.95`
- **Description** : Coef maximum quand météo = neige / conditions difficiles.

Génération des effectifs (distribution)

`staff_draw_method`

- **Type attendu** : enum
- **Valeurs possibles** :
    - `uniform`
    - `truncated_normal`
- **Description** : Méthode de tirage de l'effectif dans [borne_min, borne_max].

Taux d'hospitalisation

`taux_hospitalisation`

- **Type attendu** : float
- **Exemple** : `0.35`
- **Description** : Part des admissions qui deviennent une hospitalisation.

Usage

```
nombre_hospitalisations = round(nombre_admissions × taux_hospitalisation)
```

Saturation

`seuil_saturation_lits`

- **Type attendu** : float
- **Exemple** : `0.95`
- **Description** : À partir de ce taux d'occupation, on considère l'hôpital en saturation.

Règle

```
Si taux_occupation_lits > seuil_saturation_lits → saturation
```

Fenêtres temporelles événements

`canicule_window_days`

- **Type attendu** : int
- **Exemple** : `3`
- **Description** : Fenêtre glissante utilisée dans la détection canicule (température moyenne sur N jours).

`vague_froid_window_days`

- **Type attendu** : int
- **Exemple** : `2`
- **Description** : Fenêtre glissante utilisée dans la détection vague de froid.

`event_episode_min_days`

- **Type attendu** : int
- **Exemple** : `3`
- **Description** : Durée minimale d'un épisode événementiel pour éviter le "on/off" quotidien.

`event_episode_max_days`

- **Type attendu** : int
- **Exemple** : `21`
- **Description** : Durée maximale d'un épisode.

Probabilités d'activation (événements proxy)

Quand un événement est "probabiliste" (grippe, gastro, pollen), on paramètre :

`proba_activation_grippe`

- **Type attendu** : float
- **Exemple** : `0.30`
- **Description** : Probabilité d'activer l'épisode grippe quand les conditions sont réunies.

`proba_activation_gastro`

- **Type attendu** : float
- **Exemple** : `0.20`

- **Description** : Probabilité d'activer l'épisode gastro quand les conditions sont réunies.

`proba_activation_pollen`

- **Type attendu** : float
- **Exemple** : `0.25`
- **Description** : Probabilité d'activer l'épisode pollen quand les conditions sont réunies.

Contrôles qualité recommandés

- Unicité de `parametre`
- Cast validé (int/float/bool/enum)
- Valeurs dans les domaines (ex : `[0,1]` pour probabilités)
- Log complet des paramètres utilisés dans le run

🔗 Relations avec les autres pages

- **Utilisée par**
  - `daily_hospital_context` (bruit, taux hospitalisation, saturation)
  - `staff_variation_rules` (méthode de tirage, coefficients)
  - `event_detection_rules` (fenêtres, durées, probabilités)
  - `weather_daily_reference` (si simulation météo)
- **Ne dépend d'aucune autre table**
- C'est la **source centrale de configuration**.

Regles en plus

1️⃣ Tendances par jour de la semaine

(Base = 1.00 = jour “normal”)

📊 Nombre_Admission (toute activité hôpital)

Jour	Multiplicateur	Justification pédagogique
Lundi	1.10	Effet report week-end
Mardi	1.05	Forte activité programmée
Mercredi	1.00	Journée standard
Jeudi	1.00	Journée standard
Vendredi	0.95	Baisse actes programmés
Samedi	0.85	Activité réduite
Dimanche	0.80	Minimum hebdomadaire

👉 Règle

Nombre_Admission_jour = base_journalière × multiplicateur_jour_semaine

---

2️⃣ Répartition du personnel sur la semaine

(Effectif TOTAL journalier disponible – pas les employés globaux)

👨‍⚕️ Médecins / Infirmiers / Aides-soignants

Jour	Multiplicateur personnel
Lundi	1.00
Mardi	1.00
Mercredi	1.00

Jeudi	1.00
Vendredi	0.95
Samedi	0.85
Dimanche	0.80

👉 **Règle**

Nb_personnel_jour = Effectif_total × multiplicateur_jour

3️⃣ Effet Vacances scolaires (Paris)

Type de jour	Admissions	Personnel
Hors vacances	1.00	1.00
Vacances scolaires	0.90	0.85

Justification :

- Moins d'activité programmée
- Plus de congés → moins de personnel

4️⃣ Saisonnalité annuelle (hors événements)

Saison	Admissions	Justification
Hiver	1.15	Pathologies respiratoires
Printemps	1.00	Activité normale
Été	0.90	Départs en vacances
Automne	1.05	Reprise activité

5️⃣ Événement spécial : Épidémie de grippe

Durée typique : 6 à 10 semaines (décembre → février)

Phase	Semaines	Multiplicateur admissions
Début	1–2	1.10
Pic	3–5	1.30
Fin	6–8	1.15

👉 Personnel (avec renfort) :

Situation	Personnel
Normal	1.00
Grippe active	1.05 à 1.10

6️⃣ Événement spécial : Épidémie de gastro-entérite

Durée : 3 à 5 semaines (hiver)

Phase	Multiplicateur
Début	1.10
Pic	1.20
Fin	1.05

Impact surtout sur :

- urgences
- pédiatrie

7️⃣ Événement spécial : Canicule

Durée : 5 à 10 jours (été)

Température	Multiplicateur admissions
< 30°C	1.00
30–34°C	1.10
≥ 35°C (canicule)	1.25

Effet fort sur :

- personnes âgées
- pathologies cardio-respiratoires

8️⃣ Événement spécial : Allergies / pollen

Période : avril → juin

Intensité	Multiplicateur
Faible	1.00
Moyenne	1.05
Forte	1.10

9️⃣ Lits occupés (capacité fixe = 1800)

Hypothèse simple et bien notée :

```
Lits_occupes_jour =
min(
   1800,
   Lits_occupes_j-1
   + Admissions_jour × taux_hospitalisation
   - Sorties_jour
)
```

Paramètres pédagogiques :

- Taux hospitalisation : **20–30%** des admissions
- Durée moyenne séjour : **5 à 7 jours**