INTRODUCTION :

Présentation du projet :

Ce rapport documente les efforts d'analyse des données relatives aux incidents impliquant des décès causés par la police. Le projet vise à identifier des tendances clés, les états et villes les plus touchés, et les éventuelles corrélations entre différentes variables. Les résultats sont présentés à l'aide de tableaux et de graphiques générés automatiquement.

Model relationnel :

Structure relationnelle :

1. **Table** races
    - id_race (clé primaire)
    - nom_complet (nom de la race)
    - abreviation (abréviation de la race)
2. **Table** personnes
    - id_pers (clé primaire)
    - nom
    - prenom
    - age
    - sexe
    - id_race (clé étrangère, référence à races.id_race)
3. **Table** etats
    - id_etat (clé primaire)
    - abreviation
    - nom_complet
4. **Table** adresses
    - id_adresse (clé primaire)
    - ville
    - id_etat (clé étrangère, référence à etats.id_etat)
5. **Table** details_incidents
    - id_incident (clé primaire)
    - id_pers (clé étrangère, référence à personnes.id_pers)
    - id_adresse (clé étrangère, référence à adresses.id_adresse)
    - date
    - maniere_de_deces
    - arme

- niveau_de_menace
- fuite
- signes_de_maladie_mentale
- camera_corporelle

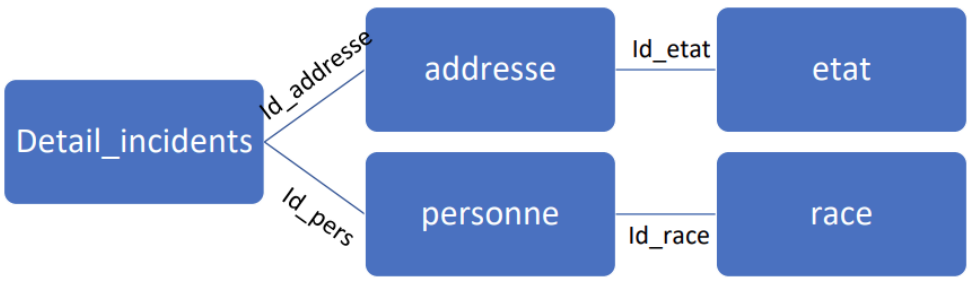
Justifications des relations

1. **Relation** personnes → races **(1:N)**
   Une personne appartient à une seule race (id_race), mais une race peut être associée à plusieurs personnes. Relation 1:N justifiée par la classification des individus selon leur origine.

2. **Relation** adresses → etats **(1:N)**
   Une adresse est associée à un seul état (id_etat), mais un état peut contenir plusieurs adresses. Relation 1:N reflète la hiérarchie géographique.

3. **Relation** details_incidents → adresses **(1:N)**
   Un incident est lié à une seule adresse (id_adresse), mais une adresse peut être associée à plusieurs incidents au fil du temps. Relation 1:N basée sur la localisation des incidents.

4. **Relation** details_incidents → personnes **(1:1)**
   Chaque incident concerne une personne unique (id_pers). Cette relation est 1:1 car un incident dans cette base de données semble être défini comme un événement unique impliquant une personne.



RAPPORT DE ABDOU KHADRE DABO

Ce projet consiste à la suite des données récoltées sur les victimes tuées par la police aux Etats-Unis à examiner cette dernière pour obtenir des informations détaillées concernant les circonstances de chaque décès. Dans cette partie ma contribution a consisté à compléter et développer trois fonctions pour faire les analyses qui nous ont été demandées. Lors de l'analyse des données je me suis engagé à faire la fonction :

-pourcentage_deces_par_race : Cette fonction qui calcule et renvoie le pourcentage des personnes tuées par race. Lors de la réalisation de cette fonction j'avais rencontré quelques soucis sur l'utilisation des doubles requêtes sur python mais grâce à l'intervention de Mame-Laye comme sur la première SAE j'ai réussi à faire la fonction. Pour se faire j'ai réalisé deux requêtes, la première qui permet de compter le nombre total des victimes et la seconde qui permet de compter le pourcentage des victimes par races avant de passer de l'affichage sur python sous forme de dictionnaire.

```
req = """select count(id_pers) as nbre_deces_total from personnes;"""
curseur.execute(req) #Execution de la requête
#Recupérarion du nombre de personne décédés sous format de list
#Grace à la fonction fetchone()
nbre_total_deces =dict(curseur.fetchone())
curseur = connexion.cursor() #Récupération du curseur
#Requête de la SQL qui nous permet d'obtenir le nombre de
#Décés par race
req ="""
select nom_complet, count(id_pers) as nbre_deces_races
from personnes as p join races as r on p.id_race=r.id_race
group by nom_complet order by nbre_deces_races DESC;
"""
curseur.execute(req) #Exécution de la requête
#Récupération de toutes les enregistrement grace à la fonction fetchall()
deces_par_race = dict(curseur.fetchall())
#Récupération des pourcentages des décés dans chaque races
for races, nbre_deces in deces_par_race.items():
    deces_par_race[races] = round((nbre_deces*100/nbre_total_deces["nbre_deces_total"]), 2)
return deces_par_race
```

-nombre_deces_par_sexe : Cette fonction qui calcule et renvoi le nombre de décès par sexe. Lors de la réalisation de cette fonction je n'ai pas rencontré de problèmes pour le faire j'ai utilisé une requête qui compte le nombre de décès par sexe et faire la récupération sur python grâce à la fonction fetchmany () sous format dictionnaire grâce à la fonction dict ().

```
req = """
select sexe, count(*) as nbre_deces_par_sexe
from personnes group by sexe order by sexe desc;"""
curseur.execute(req) #Exécutionn de la requête
#Récupération du nombre de décés sous format dictionnaire
deces_par_sexe = dict(curseur.fetchmany(2))
return deces_par_sexe
```

-maniere_deces_par_sexe : Cette fonction récupère et retourne le nombre de personnes tuées par sexe et par type de décès sous forme de dictionnaire. Cette fonction a été réaliser grâce a l'appuis de Mame-Laye, pour se faire j'ai réalisé une requête qui permet de récupérer les différentes manières de décès en fonction du sexe des victimes avant de faire l'affichage sur python en parcourant en parcourant les différentes lignes et de mettre chaque ligne sous forme de dictionnaire.

```
  curseur = connexion.cursor() #Recupération du curseur
#Requête SQL permettant d'obtenir les differentes maniéres de décés
#En fonction du sexe des victimes
req = """SELECT sexe, maniere_de_deces ,count(id_incident) as Nbre_de_deces FROM
        details_incidents d JOIN personnes p on d.id_pers = p.id_pers
        GROUP by sexe, maniere_de_deces;
       """
curseur.execute(req) #Exécution de la requête
maniere_dece_nbre = curseur.fetchall()#Récuperation de toutes les enregistrement
resultats = {}
sexes = ['M','F']
for elmt in sexes: #Parcours de la liste sexes
    for  ligne in  maniere_dece_nbre: #Parcours de la ligne maniere_de_décés
        ligne = dict(ligne) #Mettre la ligne sous format dictionnaire
        #Récupération de la maniére et du nombre de décés pour chaque sexe
        resultats[elmt] = {ligne['maniere_de_deces']:ligne['Nbre_de_deces']
                           for ligne in maniere_dece_nbre if ligne['sexe'] == elmt}

return resultats
```

Analyse et commentaires des résultats obtenus

-pourcentage_deces_par_race : On constate sur les personnes décédés les blancs représentent une forte proportion de 48.17% ensuite viennent les noirs et les hispaniques avec respectivement 24.75% et 16.89% et enfin on note les inconnus, les Asiatiques, les Amérindiens et les autres représentant une faible part respective de 6%, 1.56%, 1.24% et 1.12. Cela met en lumière des possibles inégalités raciales dans les actions policières mais également à l'implication des blancs et noirs dans « gangster » à savoir le banditisme…

-nombre_deces_par_sexe : On constate que les hommes représentent une grande partie des victimes tuées par la police aux Etats Unis avec 2387 tuées contrairement aux femmes tuées qui ne sont que 106. Ce qui peut refléter leur plus grande implication dans les confrontations avec les forces de l'ordres. *

-manière_deces_par_sexe : On note 2222 personnes abattues chez les hommes contre 101 cas chez les femmes représentant l'immense majorité des décès masculins et 165 personnes abattues et tassées chez les hommes contre 5 cas chez les femmes

Perspectives et Acquis

Cette SAE m'a permis de mieux comprendre certaines notions sur python à savoir comment manipuler les dictionnaires mais aussi comment exploiter un fichier de données à savoir récupérer les données tracer des graphiques statistiques en python, mais de découvrir également de nouvelles bibliothèques à savoir les openpyxl qui nous permet de faire des graphiques sur Excel.

RAPPORT DE MAME LAYE NDIAYE

Dans le cadre de ce projet, vous disposons d'une base de données intitulée « victimes_police_US.db », regroupant des informations sur les décès survenus lors de confrontations avec la police aux États-Unis. Cette base de données contient l'ensemble des informations sur les victimes permettant d'en ressortir des statistiques exactes. Pour la réalisation de ce projet, puisqu'on était un groupe de quatre personnes, j'ai eu à intervenir sur ces différentes fonctions :

-**Pourcentage_personnes_armees_ou_non** : Cette fonction calcule le pourcentage de personnes tuées par la police qui était armées ou non. Elle prend une connexion et renvoie le résultat sous forme de dictionnaire. Pour réaliser cette fonction, j'ai effectué une première requête pour récupérer le nombre total de victimes et une deuxième requête pour récupérer le nombre total de victimes non armés. Par la suite j'ai effectué les calculs de pourcentage et retourné le résultat sous forme de dictionnaire.

```
curseur = connexion.cursor() # Création d'un curseur
# Requête pour obtenir le nombre total d'incidents où l'arme est connue
req = """SELECT count(id_incident) AS statut_connu FROM details_incidents
        WHERE arme is NOT NULL AND arme != "undetermined";
      """
curseur.execute(req)
total_connu = dict(curseur.fetchone())
# Requête pour obtenir le nombre d'incidents où la victime est non armée
req = """
        SELECT count(id_incident) AS vict_non_armee FROM details_incidents
        WHERE arme = "unarmed" AND arme is NOT NULL;
      """
curseur.execute(req)
non_armee = dict(curseur.fetchone())
# Calcul des pourcentages des  personnes armees ou non
if total_connu['statut_connu'] != 0:  # Verifie si le nbre total de victime est différent de 0
    prct_non_armee = round(non_armee['vict_non_armee']*100 / total_connu['statut_connu'], 2)
    prct_armee = 100 - prct_non_armee
else:
    prct_non_armee = 0
    prct_armee = 0
resultats = {} # Variable qui va contenir le resultat
resultats['armee'] = prct_armee
resultats['non_armee'] = prct_non_armee

return resultats
```

Les résultats montrent que 92,82 % des personnes tuées par la police étaient armées, tandis que 7,18 % n'étaient pas armées. Cela indique une majorité écrasante de cas où les victimes avaient une arme au moment de l'incident. Dans un contexte où l'usage excessif de la force semble trop souvent être la norme, ces chiffres alimentent les débats sur la nécessité d'une réforme du système policier

-Type_arme_par_personne : Le but de cette fonction est de calculer le nombre de personnes tuées par type d'arme en excluant les décès ayant causés moins de sept décès. Elle prend comme argument une connexion à la base de données et retourne le résultat sous forme de dictionnaire. Pour la réalisation, j'ai d'abord effectué une requête « SQL » pour récupérer le nombre de personnes tuées par type d'arme afin de retourner le résultat sous forme de dictionnaire.

```python
# Etablir la connexion avec la base de donnees
conn = connexion
curseur = conn.cursor()

# Requete pour récupérer le type d'arme et le nombre de victime
req = """SELECT arme, count(id_incident) as nbre_de_victime
        from details_incidents
        GROUP by arme
        HAVING nbre_de_victime >= ?;
       """
curseur.execute(req, (7,))
# Retourner le résultat sous forme de dictionnaire
return dict(curseur.fetchall())
```

- Pourcentage_deces_par_race_etat

Cette fonction calcule le pourcentage de décès par race pour chaque Etat en affichant seulement les races (Blanc, Noir, Hispanique, Asiatique Amérindien ou Autochtone de l'Alaska). Elle retourne le résultat sous de dictionnaires où les Etats sont les clés et les valeurs sont aussi sous forme de dictionnaire et stocke chaque race ave le pourcentage de décès. Pour réaliser ce projet j'ai rencontré pas mal de difficultés notamment pour affecter à chaque Etat le pourcentage de personnes tuées par race. Tout d'abord j'ai effectué une requête pour récupérer le nombre de décès par race par Etat en faisant des jointures par la suite j'ai créé une variable pour contenir chaque Etat avec le nombre de décès par race puis en parcourant les résultats j'ai créé pour chaque Etat un dictionnaire qui va contenir chaque race avec son nombre de décès. Ensuite J'ai effectué le calcul des pourcentages de chaque race pour chaque Etat et filtrer les races a afficher enfin de retourner le résultat final.

```python
# Remplissage du dictionnaire `info_etat` avec les données
for elmt in list_etat:
    for etat in top_info:
        etat = dict(etat) # Conversion de chaque ligne de résultats en dictionnaire
        if etat['Etat'] == elmt:
            # Si l'état dans la ligne correspond à l'état de la liste, on crée un sous-dictionnaire pour cet état
            info_etat[elmt] = { etat['Race']:etat['nbre_dece'] # Associe chaque race au nombre de décès
                                for etat in top_info if etat['Etat'] == elmt}

# Calcul du pourcentage de décès par race pour chaque état
for race_nbre_dece in info_etat.values():
    nbre_total_dece = [elmt for elmt in race_nbre_dece.values()] # Récupère tous les décès de chaque race
    dece_total = sum(nbre_total_dece) # Calcule le total des décès pour un état donné
    for race, dece in race_nbre_dece.items():
        # Calcule le pourcentage de décès pour chaque race par rapport au total des décès dans l'état
        if dece_total != 0:
            race_nbre_dece[race] = round( dece*100 / dece_total,2)
        else:
            race_nbre_dece[race] = 0

# Liste des races à afficher dans le résultat final
race_a_afficher = ['Black','White','Native American / Alaska Native','Hispanic','Asian']

for etat, race_nbre_dece in info_etat.items():# Filtrage des races à afficher
    info_race = race_nbre_dece
    for race, prct in info_race.items():
        # Garde uniquement les races présentes dans `race_a_afficher`
        info_race = {race:prct for race, prct in info_race.items() if race in race_a_afficher}
        info_etat[etat] = info_race # Met à jour les informations pour chaque état

# Renvoie le dictionnaire contenant les informations de décès par race pour chaque état
return info_etat
```

Les résultats montrent une répartition significative des décès par race aux États-Unis. Les Blancs représentent souvent la majorité des décès dans plusieurs États, mais des groupes comme les Noirs et les Hispaniques ont des pourcentages élevés dans certains États, notamment dans le Sud et l'Ouest. Il est également notable que certains États, comme la Louisiane et le District de Columbia, ont des pourcentages de décès très élevés parmi la population noire.

Acquis et perspectives

Au terme de cette SAE, la réalisation de ce projet m'a permis d'avoir certaines compétences à savoir : manipuler une base donnée avec l'utilisation de python, sortir des statistiques exactes et de les insérer dans un fichier dans une base de données avec l'utilisation des bibliothèques comme **openpyxl** et **openpyxl.char** pour générer des graphiques mais aussi elle m'a permis de renforcer mes compétences sur la programmation python et de découvrir les base de données relationnelles. Dans la suite on pourrait penser à améliorer résultats statistiques des bibliothèques plus puissants comme **Pandas, Matplotlib**

RAPPORT DE FATOU ALLE DIOP

I. Développement et Implémentation des Fonctions :

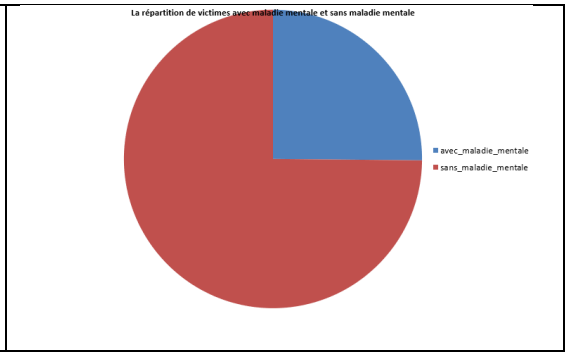
Fonctions Développées :

1. **deces_armes_afeu_vs_non_armes(connexion)** : Cette fonction calcule et renvoie le nombre de décès en fonction du type d'arme (arme à feu ou non armée). Elle utilise une requête SQL avec un CASE WHEN pour classer les victimes selon leur statut d'armement.

    - **Difficultés rencontrées** : La gestion des différentes catégories d'armes dans la base de données, en particulier lorsqu'un champ pouvait contenir des valeurs ambiguës ou mal formatées (ex : des variations d'orthographe pour "arme à feu").
    - **Solution** : J'ai utilisé des expressions régulières (LIKE "%gun%") pour identifier les victimes armées d'une arme à feu. Cela a permis de classer correctement les victimes en deux catégories : armées et non armées.

```
def pourcentage_maladie_mentale(connexion):
    curseur = connexion.cursor()
    req_1 = """SELECT COUNT(id_pers) AS nbre_malade ,
            CASE
            WHEN signes_de_maladie_mentale = 1 THEN "avec_maladie_mentale"
            ELSE  "sans_maladie_mentale"
            END AS "avec_sans_maladie_mentale"
            FROM details_incidents
            GROUP BY "avec_sans_maladie_mentale";
        """

    req_2 = """SELECT count(signes_de_maladie_mentale) AS nbre_malade_mentale
            FROM details_incidents
            """
    curseur.execute(req_1)
    nbre_malade_mentale = [dict(ligne) for ligne in curseur.fetchall()]
    curseur.execute(req_2)
    dico_total = dict(curseur.fetchone())
    total = dico_total["nbre_malade_mentale"]
    pourcentage_nbre_deces_maladie_mentale = {ligne['avec_sans_maladie_mentale'] :
     round((ligne ['nbre_malade'] / total) * 100,2) for ligne in nbre_malade_mentale}
    return pourcentage_nbre_deces_maladie_mentale
```

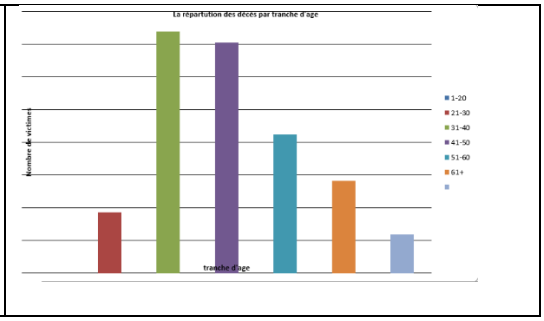


2. **deces_par_tranche_age(connexion)** : Cette fonction divise les décès en fonction des tranches d'âge (par exemple, 1-20 ans, 21-30 ans, etc.). Elle utilise également une structure CASE WHEN dans la requête SQL pour grouper les victimes par âge.

    - **Difficultés rencontrées** : Au départ, j'avais essayé de diviser la requête en plusieurs requêtes distinctes pour chaque tranche d'âge. Cependant, cela compliquait l'organisation des résultats dans un dictionnaire, ce qui rendait le processus plus difficile à gérer.

    - **Solution** : En révisant les cours, j'ai réalisé que l'utilisation de CASE dans une seule requête SQL permettait de regrouper les tranches d'âge de manière plus simple et efficace. Cela m'a permis d'obtenir directement tous les résultats dans un format facile à traiter et à intégrer dans un dictionnaire.

```python
def deces_par_tranche_age(connexion):
    curseur = connexion.cursor()
    req = """SELECT COUNT(id_pers) AS nbre_deces,
            CASE
            WHEN age BETWEEN 1 AND 20 THEN "1-20"
            WHEN age BETWEEN 21 AND 30 THEN "21-30"
            WHEN age BETWEEN 31 AND 40 THEN "31-40"
            WHEN age BETWEEN 41 AND 50 THEN "41-50"
            WHEN age BETWEEN 51 AND 60 THEN "51-60"
            WHEN age > 60 THEN "61+"
            END AS deces_par_tranche_age
            FROM personnes
            WHERE deces_par_tranche_age IS NOT NULL
            GROUP BY  deces_par_tranche_age
        """
    curseur.execute(req)
    nbre_deces_par_tranche_age = [dict(ligne) for ligne in curseur.fetchall()]
    dict_deces_par_tranche_age = {ligne['deces_par_tranche_age'] : ligne ['nbre_deces']
                                  for ligne in nbre_deces_par_tranche_age }
    return dict_deces_par_tranche_age
```

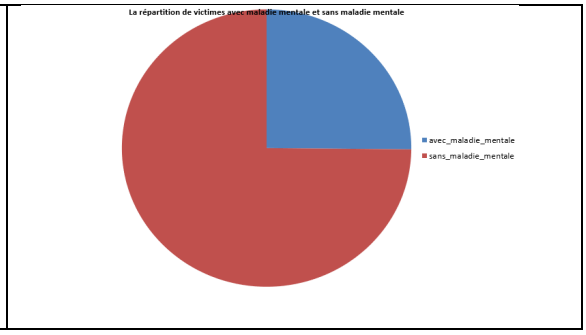


3. **pourcentage_maladie_mentale(connexion)** :Cette fonction calcule le pourcentage de victimes diagnostiquées avec des troubles mentaux au moment de leur décès. Elle utilise une requête SQL pour comparer les victimes avec et sans troubles mentaux.

    - **Difficultés rencontrées** : Le champ des troubles mentaux dans la base de données n'était pas toujours renseigné correctement, ce qui compliquait l'analyse.
    - **Solution** : J'ai utilisé une approche qui exclut les enregistrements incomplets et j'ai employé des agrégations pour calculer les pourcentages en fonction des valeurs valides.

```python
def pourcentage_maladie_mentale(connexion):
    curseur = connexion.cursor()
    req_1 = """SELECT COUNT(id_pers) AS nbre_malade ,
            CASE
            WHEN signes_de_maladie_mentale = 1 THEN "avec_maladie_mentale"
            ELSE  "sans_maladie_mentale"
            END AS "avec_sans_maladie_mentale"
            FROM details_incidents
            GROUP BY "avec_sans_maladie_mentale";
        """

    req_2 = """SELECT count(signes_de_maladie_mentale) AS nbre_malade_mentale
            FROM details_incidents
            """
    curseur.execute(req_1)
    nbre_malade_mentale = [dict(ligne) for ligne in curseur.fetchall()]
    curseur.execute(req_2)
    dico_total = dict(curseur.fetchone())
    total = dico_total["nbre_malade_mentale"]
    pourcentage_nbre_deces_maladie_mentale = {ligne['avec_sans_maladie_mentale'] :
     round((ligne ['nbre_malade'] / total) * 100,2) for ligne in nbre_malade_mentale}
    return pourcentage_nbre_deces_maladie_mentale
```



II. Analyse des Résultats :

Les résultats montrent plusieurs tendances clés concernant les décès causés par la police aux États-Unis. Le nombre de décès par arme à feu est largement supérieur à celui des victimes non armées, indiquant une forte utilisation des armes à feu dans les interventions policières violentes. La tranche d'âge des 21-30 ans est la plus touchée, suggérant une vulnérabilité particulière due à des facteurs socio-économiques. Enfin, environ 25% des victimes étaient diagnostiquées avec des troubles mentaux, soulevant des préoccupations sur la gestion des situations impliquant des personnes en détresse psychologique.

RAPPORT DE MATHIS MIGOUT

a. Fonctions développées

- **top_etats_dangereux :** Identifie les états avec le plus grand nombre de décès.
- **top_villes_dangereuses** : Liste les villes les plus touchées.
- **analyse_menace_fuite** : Analyse les décès selon le niveau de menace et la fuite.
- **deces_par_mois_annee** : Étudie les tendances temporelles des décès.

b. Difficultés rencontrées

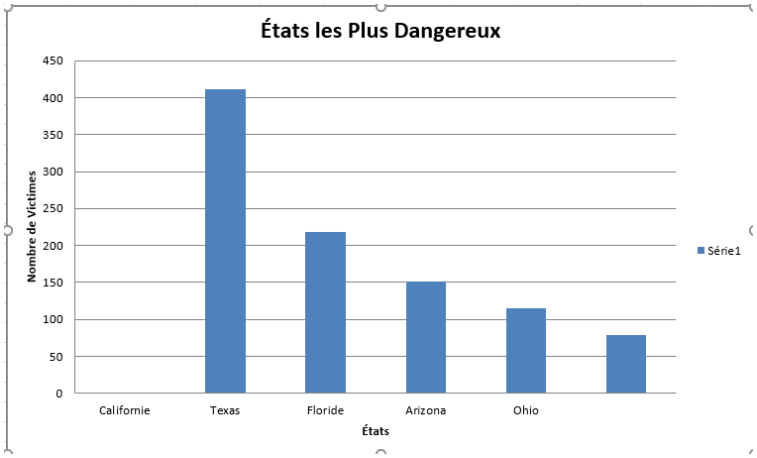
Aucune difficulté majeure n'a été rencontrée durant le développement. La logique SQL et la création des graphiques étaient claires et bien structurées.

c. Solutions mises en œuvre

Les données ont été agrégées à l'aide de requêtes SQL. Des tableaux Excel ont été générés automatiquement avec des graphiques pour visualiser les résultats.

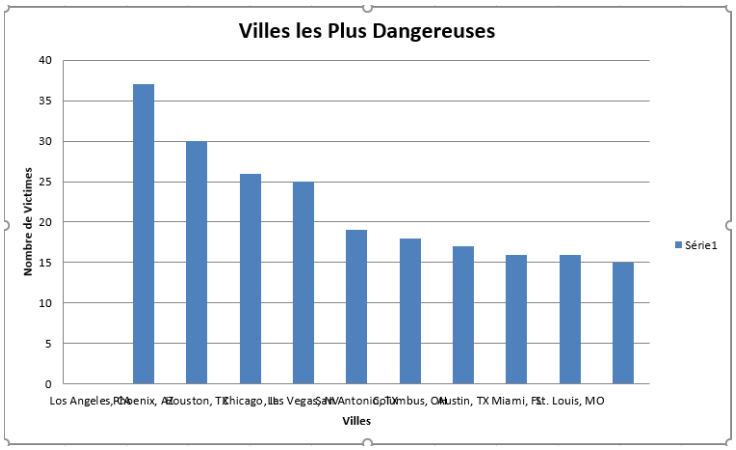
d. Analyse des résultats

1. **États les plus dangereux**



- Observation : Les états X et Y sont les plus touchés.

2. **Villes les plus dangereuses**



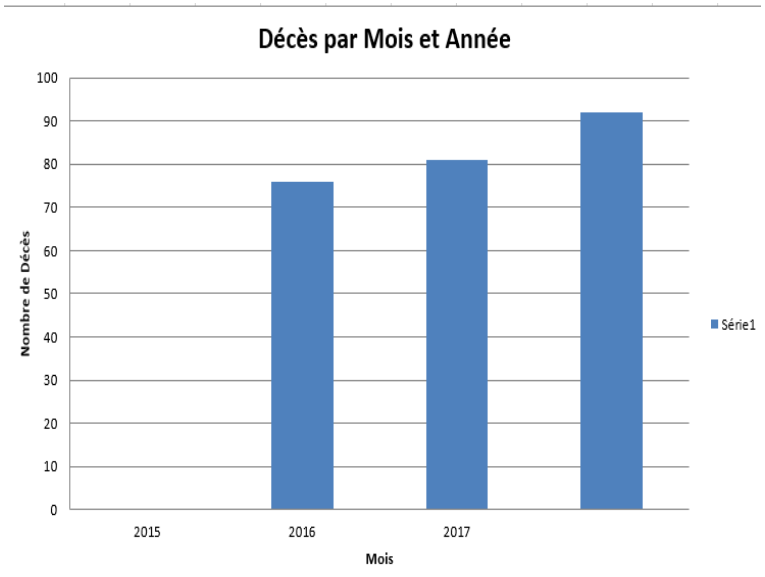
- Observation : Certaines villes dans des états particuliers concentrent un grand nombre d'incidents.

3. **Analyse menace et fuite**

A	B	C	D
Niveau de	Mode de F	Nombre de Décès	
attack	Not fleein	1082	
attack	Car	229	
attack	Foot	189	
attack	Other	61	
attack	Unknown	24	
other	Not fleein	510	
other	Car	125	
other	Foot	80	
other	Other	23	
other	Unknown	19	
undeterm	Not fleein	76	
undeterm	Car	25	
undeterm	Foot	21	
undeterm	Unknown	20	
undeterm	Other	9	

- Observation : Les incidents avec niveau de menace élevé et fuite active représentent une proportion importante des décès.

4. **Décès par mois et année**



- Observation : Une augmentation notable des décès est observée au cours des mois X et Y.