SQL QUERIES

1) Give branchwise Case Clearance Rate for the year 2024.

```
CREATE VIEW branch_cases AS

SELECT branch_id, case_id, Status

FROM personnel NATURAL JOIN assigned_case NATURAL JOIN cases;


SELECT r1.branch_id, (100*r1.closed_cases::FLOAT/r2.total_cases)::NUMERIC(4,2) AS
Case_Clearance_Rate

FROM

        (SELECT branch_id, COUNT(case_id) AS closed_cases

         FROM branch_cases WHERE status = 'Closed' AND case_id LIKE '%2024%'

         GROUP BY branch_id) AS r1

         NATURAL JOIN

        (SELECT branch_id, COUNT(case_id) AS total_cases

         FROM branch_cases WHERE case_id LIKE '%2024%'

         GROUP BY branch_id) AS r2;
```

2) Give efficiency of personnel till year 2023. We define efficiency as total number of solved cases by personnel divided by the total number of cases.

```
WITH

closed_cases AS (

        SELECT ac.personnel_id, COUNT(c.case_id) AS number_of_solved_cases

        FROM assigned_case AS ac

        NATURAL JOIN cases AS c

        WHERE c.status = 'Closed' AND

        EXTRACT(YEAR FROM c.reporting_time) <> 2024

        GROUP BY ac.personnel_id),
    open_cases AS (
        SELECT ac.personnel_id, COUNT(c.case_id) AS total_number_of_cases
        FROM assigned_case AS ac
        NATURAL JOIN cases AS c
        WHERE EXTRACT(YEAR FROM c.reporting_time) <> 2024
        GROUP BY ac.personnel_id)

    SELECT personnel_id, 100 * (closed_cases.number_of_solved_cases::FLOAT /
open_cases.total_number_of_cases)::NUMERIC(5,2) AS personnel_efficiency
    FROM closed_cases NATURAL JOIN open_cases;
```

3) Calculate branch-wise excess budget.

```
    WITH r1 AS (
            SELECT inv.branch_id, SUM(inv.stock * it.Cost) AS Cost
            FROM (inventory AS inv NATURAL JOIN items AS it)
            GROUP BY inv.branch_id
            ORDER BY Cost DESC),
         r2 AS
            (SELECT branch_id, SUM(salary) AS total_salary
            FROM personnel
            GROUP BY branch_id),
         r3 AS
             (SELECT branch_id, SUM(r1.cost + r2.total_salary) AS total_cost
             FROM r1 NATURAL JOIN r2
             GROUP BY branch_id)
        SELECT branch_id, (b.budget - r3.total_cost) AS extra_budget
        FROM branch AS b NATURAL JOIN r3
        ORDER BY extra_budget DESC;
```

4) Calculate the average case resolution time for each branch.

```
SELECT r1.branch_id,
AVG(AGE(r2.case_end_date, r2.reporting_date)) AS avg_case_resolve_time
FROM
    (SELECT branch_id, case_id
     FROM personnel
     NATURAL JOIN assigned_case) AS r1
JOIN
    (SELECT case_id, DATE(reporting_time) AS reporting_date, case_end_date
     FROM cases
  NATURAL JOIN verdict) AS r2
ON r1.case_id = r2.case_id
GROUP BY branch_id;
```

5) Unit Coordinators with the number of cases they have solved during their years of service

```
SELECT r.personnel_id, r.name, r.location, r.unit_name, count(r.case_id) as
number_of_solved_cases
FROM
    (branch as b
     NATURAL JOIN associated_unit as au
     NATURAL JOIN unit as u
     JOIN personnel as p
     ON au.coordinator_id = p.personnel_id
     NATURAL JOIN assigned_case as ac
     NATURAL JOIN cases as c) AS r
WHERE r.status='Closed'
GROUP BY (r.personnel_id, r.name, r.location, r.unit_name)
ORDER BY number_of_solved_cases DESC;
```

6) List all the cases' case_id, case_title, associated_unit for the case that are transferred from KW70 to other branch.

```
SELECT DISTINCT c.case_id, c.case_title, p.branch_id, p.unit_id

FROM cases AS c

NATURAL JOIN assigned_case AS ac

NATURAL JOIN personnel p

WHERE (crime_location LIKE '%Kolkata%') AND branch_id <> 'KW70';
```

7) Branch Heads with the number of cases they have solved during their years of service.

```
SELECT p.personnel_id, p.name, b.location, COUNT(c.case_id) AS number_of_cases

FROM

    (branch AS b

     JOIN Personnel AS p

     ON b.head_id = p.personnel_id

     NATURAL JOIN assigned_case AS ac

     NATURAL JOIN cases AS c)

WHERE c.status='Closed'

GROUP BY (p.personnel_id, p.name, b.location)

ORDER BY number_of_cases DESC;
```

8) Check if any of the suspects of a given case is already present in the Criminal Record

```
SELECT r.case_id, r.criminal_id, r.name

FROM

    (SELECT v.case_id, cr.criminal_id, cr.name

     FROM criminal_record AS cr

     NATURAL JOIN verdict AS v) AS r

JOIN suspect AS s

ON r.case_id = s.case_id

WHERE r.name = s.name
```

9) List cases in which criminals are Non-Indians.

```
SELECT c.case_id, c.case_title
FROM
        (SELECT v.case_id
         FROM verdict AS v
         NATURAL JOIN criminal_record AS cr
         WHERE  cr.nationality <> 'Indian') AS r
JOIN cases AS C ON r.case_id = c.case_id;
```

10) Criminal involved in most cases

```
SELECT cr.criminal_id, cr.name, COUNT(v.case_id) AS number_of_cases
FROM criminal_record as cr
NATURAL JOIN verdict as v
GROUP BY cr.criminal_id, cr.name
ORDER BY number_of_cases DESC LIMIT 1;
```

11) Check if any of the witness of a given case is already present in the Criminal Record

```
SELECT r.case_id, r.criminal_id, r.name
FROM
        (SELECT v.case_id, cr.criminal_id, cr.name
         FROM criminal_record AS cr
         NATURAL JOIN verdict AS v) AS r
JOIN witness AS w
ON r.case_id = w.case_id
WHERE r.name = w.name
```

12) List the criminal(s), victim(s), suspect(s) and witnesses for a given case

```
SELECT * FROM criminal_record NATURAL JOIN verdict AS v WHERE v.case_id = 'RC-02/2021/ACE/HYD'

SELECT *  FROM Victim WHERE Case_ID = 'RC-02/2021/ACE/HYD';

SELECT *  FROM Suspect WHERE Case_ID = 'RC-02/2021/ACE/HYD';

SELECT *  FROM Witness WHERE Case_ID = 'RC-02/2021/ACE/HYD';
```

13) Branch that has solved maximum number of cases

```
SELECT b.location FROM
        (SELECT branch_id, count(case_id) as number_of_solved
         FROM assigned_case as ac
         NATURAL JOIN personnel
         GROUP BY (branch_id)
         ORDER BY number_of_solved DESC LIMIT 1) as r
NATURAL JOIN branch AS b;
```

14) List Criminals who are convicted under IPC 120B for Bribery

```
SELECT cr.criminal_id, cr.name
FROM criminal_record AS cr
NATURAL JOIN verdict AS v
NATURAL JOIN cases AS c
WHERE v.court_verdict LIKE '%IPC 120B%'
```

15) List all the cases that are unsolved from 2 years.

```
SELECT *
FROM Cases
WHERE Status = 'Open' AND
CURRENT_DATE - Reporting_Time > INTERVAL '2 years';
```

16) Branch using highest amount of money in inventory

```
SELECT inv.branch_id, SUM(inv.stock * it.cost) AS Cost
FROM inventory AS inv
NATURAL JOIN items AS it
GROUP BY inv.branch_id
ORDER BY Cost DESC LIMIT 1;
```

17) List all cases along with the court verdict details for branch 'HT50'

```
SELECT c.case_id, c.case_title, v.court_verdict
FROM verdict AS v
NATURAL JOIN cases AS c
NATURAL JOIN assigned_case
NATURAL JOIN personnel
WHERE branch_id = 'HT50';
```

18) Find branch that has maximum number of Units

```
SELECT b.location, COUNT(au.unit_id) as number_of_units
FROM branch as b
NATURAL JOIN associated_unit as au
GROUP BY b.location
ORDER BY number_of_units DESC LIMIT 1;
```

19) Determine the number of officers per branch.

```
SELECT branch_id, unit_id, COUNT(personnel_id)
FROM personnel
GROUP BY branch_id, unit_id;
```

20) Branches where the stock of inventory items is less than equal to 10 units

```
SELECT b.location, it.item_name
FROM
        (branch as b
         NATURAL JOIN inventory as inv
         NATURAL JOIN items as it)
WHERE inv.stock <= 10;
```

21) List the evidences so far found for a given case

```
SELECT *  FROM evidence WHERE case_id = 'RC-06/2024/ACE/CHE';
```

22) Give personnel id of personnel with highest salary

```
SELECT personnel_id, name, salary from personnel ORDER BY salary DESC LIMIT 1;
```

23) List unit-wise average salary of personnels

```
SELECT p.unit_id, u.unit_name, AVG(salary)::NUMERIC(8,2) AS average_salary
FROM personnel AS p
NATURAL JOIN unit AS u
GROUP BY p.unit_id, u.unit_name
ORDER BY average_salary DESC;
```

24) Average no. of cases per year for every branch (Consider the data of past 4 years).

```
SELECT p.branch_id, COUNT(c.case_id)/4 :: FLOAT AS average_number_of_cases
FROM cases AS c
NATURAL JOIN assigned_case AS ac
NATURAL JOIN personnel AS p
WHERE EXTRACT(YEAR FROM reporting_time) <> 2024
GROUP BY p.branch_id;
```

25) Give the list of retired or past personnels.

```
SELECT personnel_id, name FROM personnel WHERE service_status = ‘Inactive’;
```