**Assignment 08**

***Place your query after each question***

The city’s Crime Analysis unit has submitted the following data requests. Provide the SQL statements using subqueries to satisfy the requests. Test the statements and show execution results.

1. List the name of each officer who has reported more than the average number of crimes officers have reported.

USE CityJailSummerDB

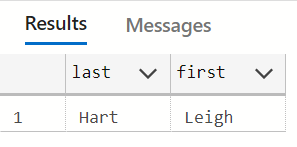
SELECT o.[last], o.[first]

FROM officers o

JOIN crime\_officers co ON o.officer\_id = co.officer\_id

GROUP BY o.[last], o.[first]

HAVING COUNT(\*) > (SELECT COUNT(\*) / COUNT(DISTINCT officer\_id) FROM crime\_officers);



1. List the names of all criminals who have committed less than average number of crimes and aren’t listed as violent offenders.

USE CityJailSummerDB

SELECT c.[last], c.[first], c.v\_status

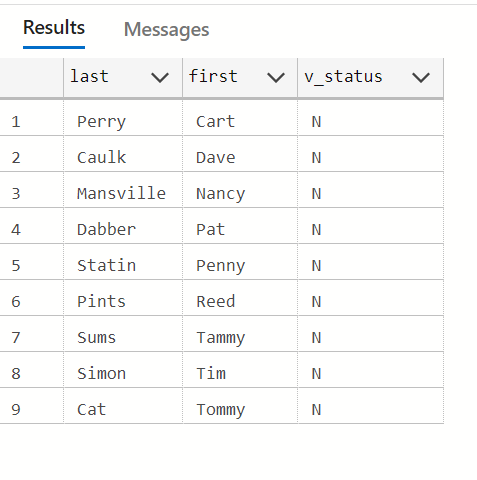
FROM criminals c

JOIN crimes cr ON c.criminal\_id = cr.criminal\_id

GROUP BY c.[last], c.[first], c.v\_status

HAVING COUNT(\*) <= (SELECT COUNT(\*) / COUNT(DISTINCT criminal\_id) FROM crimes)

AND c.v\_status = 'N';



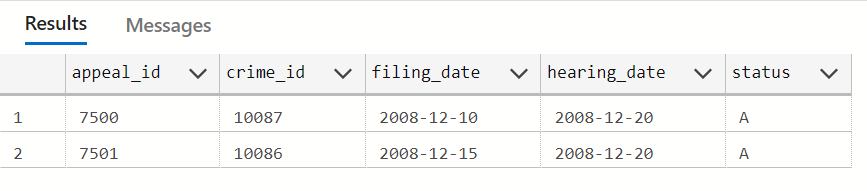
1. List appeal information for each appeal that has a less than average number of days between the filing and hearing dates.

USE CityJailSummerDB

SELECT \* FROM appeals a

WHERE DATEDIFF(day, a.filing\_date, a.hearing\_date) <

(SELECT AVG(DATEDIFF(day, a.filing\_date, a.hearing\_date))FROM appeals a);



1. List the names of probation officers who have had a less than average number of criminals assigned.

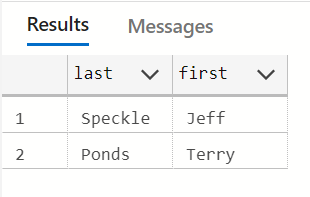
SELECT po.[last], po.[first]

FROM prob\_officers po

JOIN sentences s ON po.prob\_id = s.prob\_id

GROUP BY po.[last], po.[first]

HAVING COUNT(\*) <= (SELECT COUNT(\*) / COUNT(DISTINCT prob\_id) FROM prob\_officers);



1. List each crime that has had the highest number of appeals recorded.

USE CityJailSummerDB

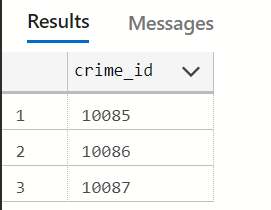
SELECT c.crime\_id

FROM crimes c

JOIN appeals co ON c.crime\_id= co.crime\_id

GROUP BY c.crime\_id

HAVING COUNT(\*) >= (SELECT COUNT(\*) / COUNT(DISTINCT crime\_id) FROM crimes);



1. List the information on crime charges for each charge that has had a fine above average and a sum paid below average.

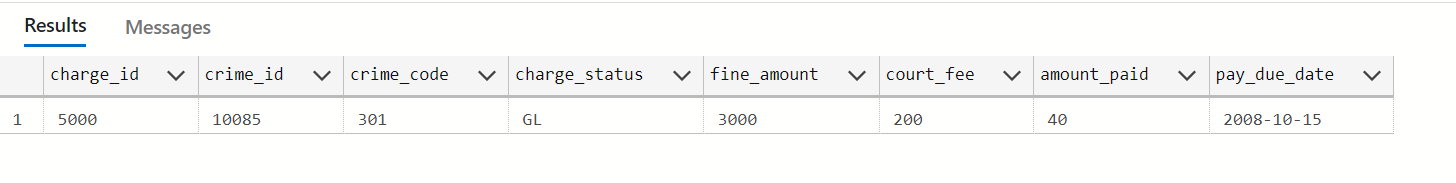
USE CityJailSummerDB

SELECT charge\_id, crime\_id, crime\_code, charge\_status, fine\_amount, court\_fee, amount\_paid, pay\_due\_date

FROM crime\_charges

WHERE fine\_amount > (SELECT AVG(fine\_amount) FROM crime\_charges)

  AND amount\_paid < (SELECT AVG(amount\_paid) FROM crime\_charges);



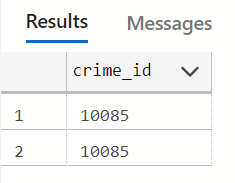
1. List the names of all criminals who have had any of the crime code charges involved in crime ID 10089.

USE CityJailSummerDB

SELECT cch.crime\_id

FROM crime\_charges cch

WHERE cch.crime\_id = '10085'



1. Use a correlated subquery to determine which criminals have had at least one probation period assigned.

USE CityJailSummerDB

SELECT c.[last], c.[first]

FROM criminals c

WHERE EXISTS (

  SELECT 1

  FROM sentences s

  WHERE s.criminal\_id = c.criminal\_id AND p\_status = 'Y'

);



1. List the names of officers who have booked the highest number of crimes. Note that more than one officer might be listed.

USE CityJailSummerDB

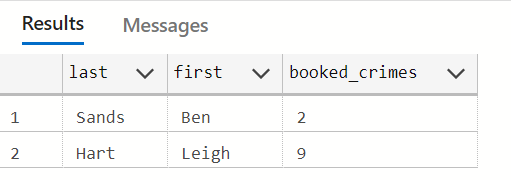
SELECT o.[last], o.[first], COUNT(co.officer\_id) AS booked\_crimes

FROM officers o

JOIN crime\_officers co ON o.officer\_id = co.officer\_id

GROUP BY o.[last], o.[first]

HAVING COUNT(\*) > (SELECT COUNT(\*) / COUNT(DISTINCT officer\_id) FROM officers);



Note: Use a MERGE statement to satisfy the following request:

1. The criminal data warehouse contains a copy of the CRIMINALS table that needs to be updated periodically from the production CRIMINALS table. The data warehouse table is named CRIMINALS\_DW. Use a single SQL statement to update the data warehouse table to reflect any data changes for existing criminals and to add new criminals.

USE CityJailSummerDB

MERGE criminals\_dw AS target

USING criminals AS SOURCE

ON (target.criminal\_id = SOURCE.criminal\_id)

WHEN MATCHED THEN

    UPDATE SET

        TARGET.last = SOURCE.last,

        TARGET.first = SOURCE.first,

        TARGET.street = SOURCE.street,

        TARGET.city = SOURCE.city,

        TARGET.state = SOURCE.state,

        TARGET.zip = SOURCE.zip,

        TARGET.phone = SOURCE.phone,

        TARGET.v\_status = SOURCE.v\_status,

        TARGET.p\_status = SOURCE.p\_status,

WHEN NOT MATCHED THEN

    INSERT(criminal\_id, last, first, street, city, state, zip, phone, v\_status, p\_status)

    VALUES(SOURCE.criminal\_id, SOURCE.last, SOURCE.first, SOURCE.street, SOURCE.city, SOURCE.state, SOURCE.zip, SOURCE.phone, SOURCE.v\_status, SOURCE.p\_status);