

Validation Queries for Given Questions

1. How many accidents occurred in NYC, Austin, and Chicago?

```
SELECT
    dl.City,
    COUNT(*) AS Accident_Count
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
WHERE
    dl.City IN ('New York', 'Austin', 'Chicago')
GROUP BY
    dl.City;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central query editor shows the following SQL query:

```
SELECT
    dl.City,
    COUNT(*) AS Accident_Count
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
WHERE
    dl.City IN ('New York', 'Austin', 'Chicago')
GROUP BY
    dl.City;
```

The query has been executed successfully, as indicated by the status bar. The Results pane shows the following data:

City	Accident_Count
Chicago	817722
New York	2062607
Austin	147750

The Object Explorer on the left shows the database structure, including the 'Fact_Accident_table' and 'Dim_Location' tables. The Properties pane on the right shows the connection details for the 'localhost (d2)' connection.

2. Which areas in the 3 cities had the greatest number of accidents?

top 3 areas in each city

```
WITH RankedAreas AS (  
    SELECT  
        dl.City,  
        dl.Address,  
        ROW_NUMBER() OVER (PARTITION BY dl.City ORDER BY COUNT(*) DESC) AS Area_Rank,  
        COUNT(*) AS Accident_Count  
    FROM  
        Fact_Accident_table fat  
    JOIN  
        Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]  
    WHERE  
        dl.City IN ('New York', 'Austin', 'Chicago')  
    GROUP BY  
        dl.City, dl.Address  
)  
SELECT  
    City,  
    Address,  
    Accident_Count  
FROM  
    RankedAreas  
WHERE  
    Area_Rank <= 3;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the execution of a SQL query. The query uses a Common Table Expression (CTE) named 'RankedAreas' to rank accident locations by count within each city. The final SELECT statement retrieves the top 3 areas for each city.

Query Results:

City	Address	Accident_Count
New York	Unknown	92072
New York	BROADWAY	19839
New York	BELT PARKWAY	17679
Austin	NOT REPORTED	10178
Austin	IH 35	3969
Austin	LAMAR	3828
Chicago	WESTERN AVE	22319
Chicago	PULASKI RD	19695
Chicago	CICERO AVE	18335

The status bar at the bottom indicates the query was executed successfully, returning 9 rows in 0.000000 seconds.

3. How many accidents resulted in just injuries?

this report needs to be generated at 2 levels, 1 -> overall, 3 -> by city

-- By city

```
SELECT
    dl.City,
    SUM(fat.Total_injury_count) AS City_Injury_Sum
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
WHERE
    fat.Total_injury_count > 0 AND fat.Total_injury_count <> -999
GROUP BY
    dl.City;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the execution of a T-SQL query. The query is as follows:

```
-- By city
SELECT
    dl.City,
    SUM(fat.Total_injury_count) AS City_Injury_Sum
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
WHERE
    fat.Total_injury_count > 0 AND fat.Total_injury_count <> -999
GROUP BY
    dl.City;
```

The Results pane at the bottom shows the output of the query:

City	City_Injury_Sum
Chicago	154967
New York	638871
Austin	94855

The status bar at the bottom indicates that the query was executed successfully on the 'localhost (16.0 RTM) d2 (66)' server, returning 3 rows in 00:00:00. The taskbar at the very bottom shows the system clock as 21:36 on 13-04-2024.

-- Overall

```
SELECT
    SUM(fat.Total_injury_count) AS Overall_Injury_Sum
FROM
    Fact_Accident_table fat
WHERE
    fat.Total_injury_count > 0 AND fat.Total_injury_count <> -999;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the following SQL query:

```
-- Overall
SELECT
    SUM(fat.Total_injury_count) AS Overall_Injury_Sum
FROM
    Fact_Accident_table fat
WHERE
    fat.Total_injury_count > 0 AND fat.Total_injury_count <> -999;
```

The 'Results' pane at the bottom shows a single row with the value 888693 under the column 'Overall_Injury_Sum'.

The 'Properties' pane on the right provides details about the connection and query execution:

- Aggregate Status:** Connection failure, Elapsed time: 00:00:00.089, Finish time: 13-04-2024 21:37:40, Name: localhost, Rows returned: 1, Start time: 13-04-2024 21:37:40, State: Open.
- Connection:** Connection name: localhost (d2).
- Connection Details:** Connection elapsed: 00:00:00.089, Connection encrypt: Not encrypted, Connection finish: 13-04-2024 21:37:40, Connection rows: 1, Connection start: 13-04-2024 21:37:40, Connection state: Open, Display name: localhost, Login name: d2, Server name: localhost.
- Name:** The name of the connection.

The status bar at the bottom indicates 'Query executed successfully.' and shows the connection details: 'localhost (16.0 RTM) | d2 (66) | Final_Project | 00:00:00 | 1 rows'.

4. How often are pedestrians involved in accidents?

this report needs to be generated at 2 levels, 1 -> overall, 3 -> by city

-- By city

```
SELECT
    dl.City,
    YEAR(dd.dt) AS Year,
    MONTH(dd.dt) AS Month,
    SUM(IIF(fat.Pedestrians_inv_acc <> -999, fat.Pedestrians_inv_acc, 0)) AS
City_Pedestrian_Accidents
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
JOIN
    Dim_Date dd ON fat.[date(SK)] = dd.[date(SK)]
WHERE
    fat.Pedestrians_inv_acc > 0 AND fat.Pedestrians_inv_acc <> -999
GROUP BY
    dl.City, YEAR(dd.dt), MONTH(dd.dt)
ORDER BY
    YEAR(dd.dt), dl.City;
```

SQLQuery2.sql - localhost.Final_Project (d2 (66)) - Microsoft SQL Server Management Studio

Object Explorer: localhost (SQL Server 16.0.10006 - d2)

Query: SQLQuery2.sql - localhost.Final_Project (d2 (66))

Results: 97 %

City	Year	Month	City_Pedestrian_Accidents
New York	2012	12	1305
New York	2012	10	982
New York	2012	7	852
New York	2012	9	902
New York	2012	8	859
New York	2012	11	1040
Chicago	2013	6	1
New York	2013	12	1173
New York	2013	11	1123
New York	2013	3	992
New York	2013	4	905
New York	2013	1	1127
New York	2013	6	977
New York	2013	2	991
New York	2013	10	1074
New York	2013	9	1024
New York	2013	8	860

Query executed successfully. localhost (16.0 RTM) d2 (66) Final_Project 00:00:00 367 rows

Properties: Connection failure, Elapsed time: 00:00:00.265, Finish time: 13-04-2024 21:39:07, Name: localhost, Rows returned: 367, Start time: 13-04-2024 21:39:07, State: Open.

-- Overall

```
SELECT
    SUM(CASE WHEN fat.Pedestrians_inv_acc > 0 THEN fat.Pedestrians_inv_acc ELSE 0
END) AS Overall_Pedestrians_Sum
FROM
    Fact_Accident_table fat
WHERE
    fat.Pedestrians_inv_acc <> -999;
```

The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The main window shows a SQL query in the 'SQLQuery2.sql' file, which is the same query as provided in the text above. The query has been executed successfully, as indicated by the status bar at the bottom: 'Query executed successfully. localhost (16.0 RTM) | d2 (66) | Final_Project | 00:00:00 | 1 rows'.

The 'Object Explorer' on the left shows the database structure for 'localhost (SQL Server 16.0.1000.6 - d2)', including various databases like 'System Databases', 'Database Snapshots', 'Chinook', 'Chinook', 'DEMO', 'Final_Project', 'fresh', 'Midterm_Project', 'new', 'nw', 'poc', 'sample', 'sample_test', 'test', 'Security', 'Server Objects', 'Replication', 'Always On High Availability', 'Management', 'Integration Services Catalogs', and 'XEEvent Profiler'.

The 'Results' pane at the bottom shows the output of the query, which is a single row with the value '138694' under the column 'Overall_Pedestrians_Sum'.

The 'Properties' pane on the right shows the 'Aggregate Status' and 'Connection' details for the current connection. The 'Aggregate Status' section shows 'Connection failure' with 'Elapsed time' of '00:00:00.100', 'Finish time' of '13-04-2024 21:45:23', 'Name' of 'localhost', 'Rows returned' of '1', 'Start time' of '13-04-2024 21:45:23', and 'State' of 'Open'. The 'Connection' section shows 'Connection name' as 'localhost (d2)', 'Connection elapsed' as '00:00:00.100', 'Connection encrypt' as 'Not encrypted', 'Connection finish' as '13-04-2024 21:45:23', 'Connection rows' as '1', 'Connection start' as '13-04-2024 21:45:23', 'Connection state' as 'Open', 'Display name' as 'localhost', 'Login name' as 'd2', and 'Server name' as 'localhost'.

5. When do most accidents happen? seasonality report

```
SELECT
    d.Year_Num,
    dl.City,
    d.Season,
    COUNT(*) AS Accident_Count
FROM
    Fact_Accident_table fat
JOIN
    Dim_Date d ON fat.[date(SK)] = d.[date(SK)]
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
GROUP BY
    d.Year_Num, dl.City, d.Season;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central query editor shows the following SQL query:

```
SELECT
    d.Year_Num,
    dl.City,
    d.Season,
    COUNT(*) AS Accident_Count
FROM
    Fact_Accident_table fat
JOIN
    Dim_Date d ON fat.[date(SK)] = d.[date(SK)]
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
GROUP BY
    d.Year_Num, dl.City, d.Season;
```

The Results pane at the bottom shows the output of the query, which is a table with 17 rows and 4 columns: Year_Num, City, Season, and Accident_Count. The data is as follows:

Year_Num	City	Season	Accident_Count
2017	Chicago	Winter	13577
2017	Austin	Spring	4005
2021	Chicago	Fall	27559
2020	New York	Fall	27050
2017	New York	Fall	59382
2023	New York	Spring	24796
2018	New York	Summer	58330
2018	Austin	Winter	3973
2018	Chicago	Spring	30963
2016	Chicago	Fall	14767
2021	New York	Spring	29532
2019	New York	Spring	55725
2015	Austin	Spring	3805
2023	Chicago	Fall	27976
2015	Chicago	Winter	4
2020	Austin	Spring	2352
2023	Chicago	Summer	28911

The right-hand side of the interface shows the Properties pane with connection details for the 'localhost (d2)' connection. The status is 'Open', and the connection was established on 13-04-2024 at 21:53:33. The message pane at the bottom indicates that the query was executed successfully, returning 131 rows.

6. How many motorists are injured or killed in accidents?

this report needs to be generated at 2 levels, 1 -> overall, 3 -> by city

– By city

```
SELECT
    dl.City,
    SUM(IIF(fat.Motorist_injured <> -999, fat.Motorist_injured, 0)) + SUM(IIF(fat.Motorist_killed <> -999,
fat.Motorist_killed, 0)) AS City_Motorists_Injured_or_Killed
FROM
    Fact_Accident_table fat
JOIN
    Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]
WHERE
    (fat.Motorist_injured > 0 OR fat.Motorist_killed > 0) AND (fat.Motorist_injured <> -999 OR
fat.Motorist_killed <> -999)
GROUP BY
    dl.City;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the execution of a SQL query. The query is designed to count the number of injured and killed motorists by city, excluding records with values of -999. The results pane shows the following data:

City	Motorists_Injured	Motorists_Killed
Austin	4047	543
Chicago	0	0
New York	459447	1261

The status bar at the bottom indicates that the query executed successfully, returning 3 rows. The right-hand pane shows the Properties window, which includes connection details and aggregate status information.

– Overall

SELECT

SUM(IIF(fat.Motorist_injured <> -999, fat.Motorist_injured, 0)) + SUM(IIF(fat.Motorist_killed <> -999, fat.Motorist_killed, 0)) AS Overall_Motorists_Injured_or_Killed

FROM

Fact_Accident_table fat;

The screenshot displays the Microsoft SQL Server Management Studio interface. The central query editor shows the following SQL query:

```
SELECT  
SUM(IIF(fat.Motorist_injured <> -999, fat.Motorist_injured, 0)) + SUM(IIF(fat.Motorist_killed <> -999, fat.Motorist_killed, 0)) AS Overall_Motorists_Injured_or_Killed  
FROM  
Fact_Accident_table fat;
```

The query has been executed successfully, as indicated by the status bar. The Results pane shows a single row with the value 465298 for the column Overall_Motorists_Injured_or_Killed.

Overall_Motorists_Injured_or_Killed
465298

The right-hand pane displays the Properties window for the connection, showing details such as Connection name (localhost (d2)), Connection state (Open), and Connection details (Connection elapsed time, Connection finish, Connection start time, Connection state, Display name, Login name, Server name).

7. Which top 5 areas in 3 cities have the most fatal number of accidents?

WITH FatalAccidentsByCity AS (

```
SELECT
    dl.City,
    dl.Address,
    COUNT(fat.No_Fatal_Acc) AS TotalFatalAccidents
```

FROM

Fact_Accident_table fat

JOIN

Dim_Location dl ON fat.[Location_id(SK)] = dl.[Location_id(SK)]

GROUP BY

dl.City, dl.Address

)

SELECT

City,

Address,

TotalFatalAccidents

FROM (

SELECT

City,

Address,

TotalFatalAccidents,

ROW_NUMBER() OVER(PARTITION BY City ORDER BY TotalFatalAccidents DESC) AS

RowNumber

FROM

FatalAccidentsByCity

) AS RankedFatalAccidents

WHERE

RowNumber <= 5;

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The central pane shows a SQL query in the 'SQLQuery2.sql' file. The query is a CTE-based query that ranks fatal accidents by city and address. The results pane at the bottom shows the output of the query, which is a table with 15 rows. The table has three columns: City, Address, and TotalFatalAccidents. The results are ordered by City and then by TotalFatalAccidents in descending order. The first five rows are for Chicago, and the next five are for Austin, and the last five are for New York. The status bar at the bottom indicates that the query was executed successfully and returned 15 rows.

City	Address	TotalFatalAccidents
Chicago	WESTERN AVE	22319
Chicago	PULASKI RD	19695
Chicago	CICERO AVE	18335
Chicago	ASHLAND AVE	17795
Chicago	HALSTED ST	15851
Austin	NOT REPORTED	10178
Austin	IH 35	3969
Austin	LAMAR	3828
Austin	MOPAC	3697
Austin	PARMER	1775
New York	Unknown	92072
New York	BROADWAY	19839
New York	BELT PARKWAY	17679
New York	ATLANTIC AVENUE	17639
New York	3 AVENUE	14238

8. Time-based analysis of accidents

Time of the day, day of the week, weekdays or weekends

```
SELECT
    D_Date.Year_Num AS [Year],
    D_Date.Is_Weekend AS [Weekend],
    D_Date.Date_Str AS [DateString],
    D_Time.Time_of_the_day AS [TimeOfDay],
    COUNT(F_Accident.Case_id) AS [AccidentCount]
FROM
    Fact_Accident_table AS F_Accident
    INNER JOIN Dim_Date AS D_Date ON F_Accident.[date(SK)] = D_Date.[date(SK)]
    INNER JOIN Dim_Time AS D_Time ON F_Accident.[Time_id(SK)] = D_Time.[Time_id(SK)]
GROUP BY
    D_Date.Year_Num,
    D_Date.Is_Weekend,
    D_Date.Date_Str,
    D_Time.Time_of_the_day
ORDER BY
    [Year],
    [Weekend],
    [DateString],
    [TimeOfDay];
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows a SQL query for time-based analysis of accidents. The query selects Year, Weekend status, DateString, TimeOfDay, and AccidentCount, grouped by Year, Weekend, DateString, and TimeOfDay. The results pane at the bottom shows 17 rows of data for the year 2012, categorized by weekend status, date string, and time of day.

Year	Weekend	DateString	TimeOfDay	AccidentCount
2012	N	Friday	Afternoon	5264
2012	N	Friday	Evening	3782
2012	N	Friday	Morning	4662
2012	N	Friday	Night	2474
2012	N	Monday	Afternoon	4604
2012	N	Monday	Evening	3369
2012	N	Monday	Morning	4360
2012	N	Monday	Night	1890
2012	N	Thursday	Afternoon	4736
2012	N	Thursday	Evening	3471
2012	N	Thursday	Morning	4433
2012	N	Thursday	Night	2020
2012	N	Tuesday	Afternoon	4714
2012	N	Tuesday	Evening	3482
2012	N	Tuesday	Morning	4465
2012	N	Tuesday	Night	1899
2012	N	Wednesday	Afternoon	4761

The right-hand pane shows the Properties window for the current connection (localhost (d2)), displaying connection details such as connection name, connection details, and connection state.

9. Fatality analysis

Are pedestrians killed more often than road users?

```
SELECT
    YEAR(dd.dt) AS Year,
    SUM(IIF(fat.Pedestrians_inv_acc <> -999, fat.Pedestrians_inv_acc, 0)) AS PedestrianFatalities,
    SUM(IIF(fat.No_Road_Users <> -999, fat.No_Road_Users, 0)) AS RoadUserFatalities
FROM
    Fact_Accident_table fat
JOIN
    Dim_Date dd ON fat.[date(SK)] = dd.[date(SK)]
GROUP BY
    YEAR(dd.dt)
ORDER BY
    Year;
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows a SQL query being executed. The query is a SELECT statement that calculates the number of pedestrian and road user fatalities by year, using the Fact_Accident_table and Dim_Date tables. The query is executed successfully, and the results are displayed in the Results pane at the bottom. The Results pane shows a table with three columns: Year, PedestrianFatalities, and RoadUserFatalities. The data shows a general downward trend in fatalities over the years, with a significant drop in 2024.

Year	PedestrianFatalities	RoadUserFatalities
2012	5640	65
2013	12062	120
2014	11147	167
2015	10302	180
2016	11536	136
2017	12966	188
2018	14275	149
2019	13773	165
2020	8821	227
2021	9869	224
2022	11509	201
2023	11853	207
2024	2841	40

The Properties pane on the right shows the connection details for the 'localhost (d2)' connection. The connection is successful, and the query is executed successfully. The status bar at the bottom indicates that the query was executed successfully and returned 13 rows.

10. What are the most common factors involved in accidents?

SELECT TOP 10

DCF.Contributing_Factor_Description,

COUNT(*) AS 'Number of Accidents'

FROM Fact_Accident_table FAT

JOIN Bridge_Collision_Contributing_Factor BCCF ON FAT.Fact_Accident_id = BCCF.Fact_Accident_id

JOIN Dim_Contributing_Factor DCF ON BCCF.[Dim_Contributing_Factor(SK)] =

DCF.[Dim_Contributing_Factor(SK)]

GROUP BY DCF.Contributing_Factor_Description

ORDER BY COUNT(*) DESC;

The screenshot displays the Microsoft SQL Server Management Studio interface. The central query editor contains the following SQL code:

```
SELECT TOP 10
DCF.Contributing_Factor_Description,
COUNT(*) AS 'Number of Accidents'
FROM Fact_Accident_table FAT
JOIN Bridge_Collision_Contributing_Factor BCCF ON FAT.Fact_Accident_id = BCCF.Fact_Accident_id
JOIN Dim_Contributing_Factor DCF ON BCCF.[Dim_Contributing_Factor(SK)] = DCF.[Dim_Contributing_Factor(SK)]
GROUP BY DCF.Contributing_Factor_Description
ORDER BY COUNT(*) DESC;
```

The Results pane at the bottom shows the output of the query, which is a table with two columns: Contributing_Factor_Description and Number of Accidents. The data is as follows:

Contributing_Factor_Description	Number of Accidents
Unknown	2181432
Driver Inattention	448733
UNABLE TO DETERMINE	425937
other	338746
Followed Too Closely	213935
Failed to Drive in Single Lane	174378
FAILING TO YIELD RIGHT-OF-WAY	104173
Unsafe Speed	98160
Fatigued or Asleep	62667
Backed without Safety	82185

The right-hand side of the interface shows the Solution Explorer with a project named 'SQLQuery2.sql' and the Properties window displaying connection details for 'localhost (d2)'.

11. Using Austin and NYC datasets, Create a visualization to show a number of incidents that involved more than 2 vehicles. Show this data as a comparison between these 2 cities.

```
SELECT
    'Austin' AS Case_Type,
    COUNT(*) as NumberOfCasesWithMoreThan2Vehicles
FROM (
    SELECT
        f.Case_id
    FROM
        Fact_Accident_table f
    JOIN
        Bridge_Vehicle_involved bvi ON f.Fact_Accident_id = bvi.Fact_Accident_id
    GROUP BY
        f.Case_id
    HAVING
        COUNT(bvi.Dim_Vehicle) > 2 AND
        f.Case_id LIKE 'A%'
) AS SubqueryA
```

UNION ALL

```
SELECT
    'New York' AS Case_Type,
    COUNT(*) as NumberOfCasesWithMoreThan2Vehicles
FROM (
    SELECT
        f.Case_id
    FROM
        Fact_Accident_table f
    JOIN
        Bridge_Vehicle_involved bvi ON f.Fact_Accident_id = bvi.Fact_Accident_id
    GROUP BY
        f.Case_id
    HAVING
        COUNT(bvi.Dim_Vehicle) > 2 AND
        f.Case_id LIKE 'N%'
) AS SubqueryN;
```


SQLQuery2.sql - localhost.Final_Project (d2 (66)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

localhost (SQL Server 16.0.1000.6 - d2)

- Databases
 - System Databases
 - Database Snapshots
 - Chinook
 - Chinook
 - DEMO
 - Final_Project
 - fresh
 - Midterm_Project
 - new
 - nw
 - poc
 - sample
 - sample_test
 - test
 - Security
 - Server Objects
 - Replication
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - XEvent Profiler

SQLQuery2.sql - loc...al_Project (d2 (66))

```
UNION ALL
SELECT
    'New York' AS Case_Type,
    COUNT(*) as NumberOfCasesWithMoreThan2Vehicles
FROM (
    SELECT
        f.Case_id
    FROM
        Fact_Accident_table f
    JOIN
        Bridge_Vehicle_involved bvi ON f.Fact_Accident_id = bvi.Fact_Accident_id
    GROUP BY
        f.Case_id
    HAVING
        COUNT(bvi.Dim_Vehicle) > 2 AND
        f.Case_id LIKE 'NS'
) AS SubqueryM
```

Results

Case_Type	NumberOfCasesWithMoreThan2Vehicles
Austin	4227
New York	14044

Messages

Query executed successfully.

localhost (16.0 RTM) | d2 (66) | Final_Project | 00:00:00 | 2 rows

Ready

Type here to search

8°C Mostly cloudy

22:56 13-04-2024

Solution Explorer

Search Solution Explorer (Ctrl+Q)

- Solution 'Solution1' (0 projects)
- Miscellaneous Files
- SQLQuery2.sql

Properties

Current connection parameters

Aggregate Status

Connection failure

Elapsed time 00:00:00.943

Finish time 13-04-2024 22:56:49

Name localhost

Rows returned 2

Start time 13-04-2024 22:56:48

State Open

Connection

Connection name localhost (d2)

Connection Details

Connection elapsed 00:00:00.943

Connection encrypt Not encrypted

Connection finish 13-04-2024 22:56:49

Connection rows 2

Connection start t 13-04-2024 22:56:48

Connection state Open

Display name localhost

Login name d2

Server name localhost

Name

The name of the connection.