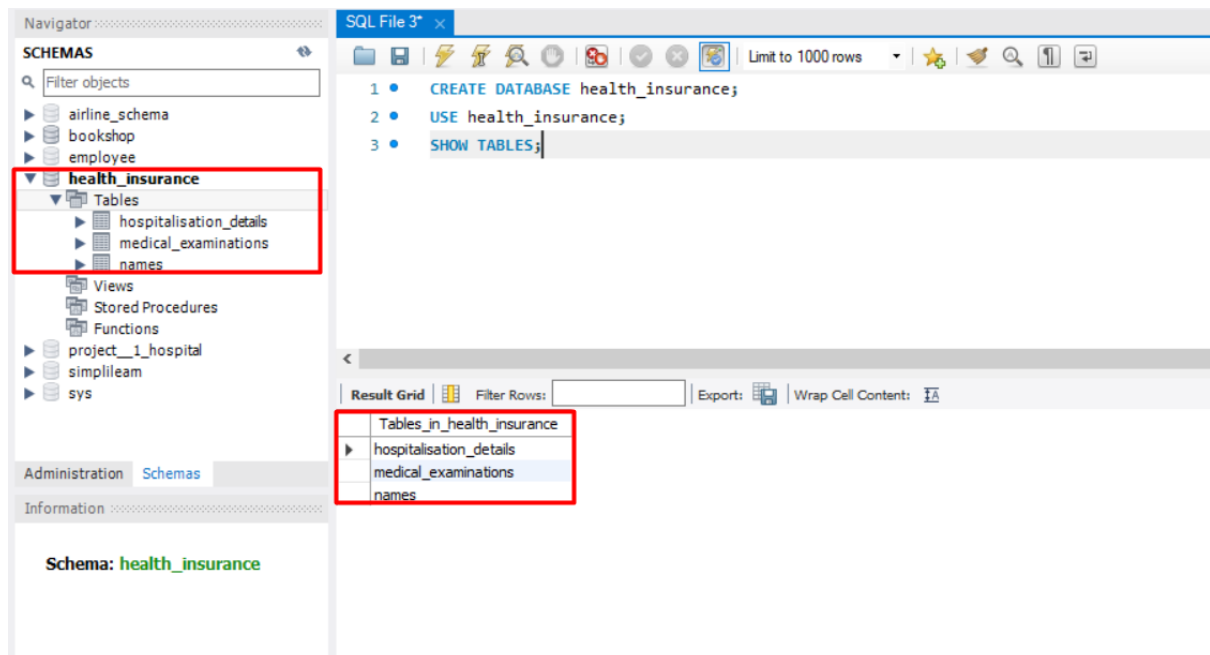


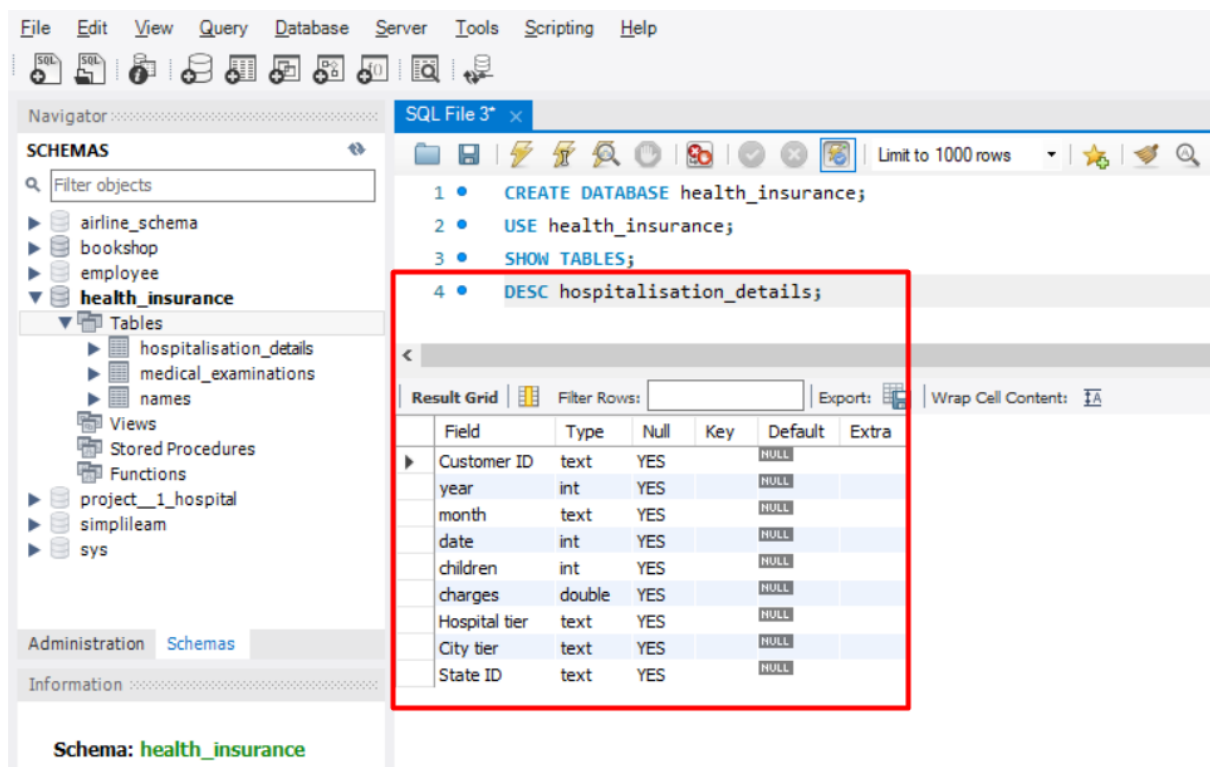
1. Created the database and imported the tables



2. Checking the columns in the table 'hospitalisation\_details'

Code: DESC hospitalisation\_details;

OUTPUT:



3. Checking the columns in the table 'medical\_examinations'

Code: DESC medical\_examinations;

OUTPUT:

SQL File 3\* x

1 • CREATE DATABASE health\_insurance;  
2 • USE health\_insurance;  
3 • SHOW TABLES;  
4 • DESC hospitalisation\_details;  
5 • DESC medical\_examinations;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Field	Type	Null	Key	Default	Extra
▶	Customer ID	text	YES		NULL	
	BMI	double	YES		NULL	
	HBA1C	double	YES		NULL	
	Heart Issues	text	YES		NULL	
	Any Transplants	text	YES		NULL	
	Cancer history	text	YES		NULL	
	NumberOfMajorSurgeries	text	YES		NULL	
	smoker	text	YES		NULL	

4. Checking the columns in the table 'names'

Code: DESC names;

OUTPUT

SQL File 3\* x

2 • USE health\_insurance;  
3 • SHOW TABLES;  
4 • DESC hospitalisation\_details;  
5 • DESC medical\_examinations;  
6 • DESC names;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Field	Type	Null	Key	Default	Extra
▶	Customer ID	text	YES		NULL	
	name	text	YES		NULL	

## 5. Checking the values in 'hospitalisation\_details' table

CODE: `SELECT * FROM hospitalisation_details;`

OUTPUT:

The screenshot shows the SQL Enterprise Manager interface. The SQL window contains the following queries:

```

3 * SHOW TABLES;
4 * DESC hospitalisation_details;
5 * DESC medical_examinations;
6 * DESC names;
7 * SELECT * FROM hospitalisation_details;

```

The Results pane displays the output of the query `SELECT * FROM hospitalisation_details;`. The results are shown in a table with the following columns: Customer ID, year, month, date, children, charges, Hospital tier, City tier, and State ID. The table contains 1000 rows of data.

The Output pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
42	11:04:30	SHOW TABLES	3 row(s) returned	0.000 sec / 0.000 sec
43	11:06:45	DESC hospitalisation_details	9 row(s) returned	0.000 sec / 0.000 sec
44	12:06:50	DESC medical_examinations	8 row(s) returned	0.000 sec / 0.000 sec
45	12:08:52	DESC names	2 row(s) returned	0.000 sec / 0.000 sec
46	12:11:08	SELECT * FROM hospitalisation_details LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

## 6. Checking the values in 'medical\_examinations' table

CODE: `SELECT * FROM medical_examinations;`

OUTPUT:

The screenshot shows the SQL Enterprise Manager interface. The SQL window contains the following queries:

```

4 * DESC hospitalisation_details;
5 * DESC medical_examinations;
6 * DESC names;
7 * SELECT * FROM hospitalisation_details;
8 * SELECT * FROM medical_examinations;

```

The Results pane displays the output of the query `SELECT * FROM medical_examinations;`. The results are shown in a table with the following columns: Customer ID, BMI, HBA1C, Heart Issues, Any Transplants, Cancer history, NumberOfMajorSurgeries, and smoker. The table contains 1000 rows of data.

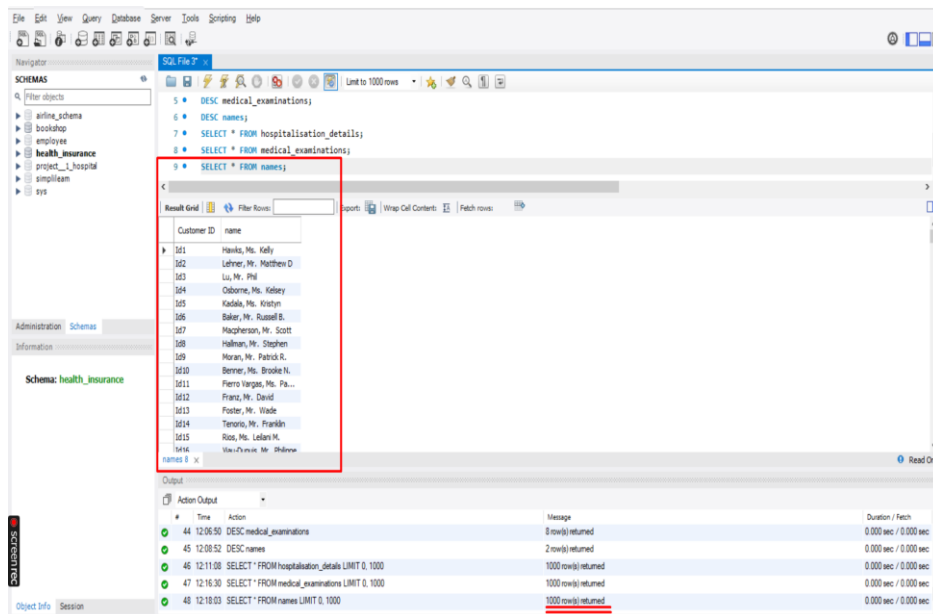
The Output pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
43	11:06:45	DESC hospitalisation_details	9 row(s) returned	0.000 sec / 0.000 sec
44	12:06:50	DESC medical_examinations	8 row(s) returned	0.000 sec / 0.000 sec
45	12:08:52	DESC names	2 row(s) returned	0.000 sec / 0.000 sec
46	12:11:08	SELECT * FROM hospitalisation_details LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
47	12:16:30	SELECT * FROM medical_examinations LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

## 7. Checking the values in 'names' table

CODE:

OUTPUT:



TASK: 1

STEP 1: Merge the two tables by first identifying the columns in the data tables

CODE:

CREATE TABLE merged\_data AS

SELECT H.\*,

M.BMI,

M.HBA1C,

M.`Heart Issues`,

M.`Any Transplants`,

M.`Cancer history`,

M.`NumberOfMajorSurgeries`,

M.smoker

FROM hospitalisation\_details AS H

JOIN `medical\_examinations` AS M ON H.`Customer ID` = M.`Customer ID`;

SELECT \* FROM merged\_data;

OUTPUT:

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'health\_insurance' schema with various tables. The central pane shows the SQL script for creating the 'merged\_data' table and selecting data from it. The right pane displays the 'Result Grid' with 10 columns: Customer ID, year, month, date, children, charges, Hospital tier, City tier, State ID, BMI, HBA1C, Heart Issues, Any Transplants, Cancer history, NumberOfMajorSurgeries, and smoker. The data is limited to 1000 rows.

Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker
Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	R.1013	17.58	4.51	No	No	No	1	No
Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	R.1013	17.6	4.39	No	No	No	1	No
Id2333	1993	Jun	30	0	600	tier - 2	tier - 1	R.1013	16.47	6.35	No	No	Yes	1	No
Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	R.1013	17.7	6.28	No	No	No	1	No
Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	R.1013	22.34	5.57	No	No	No	1	No
Id2330	2001	Nov	20	0	646.14	tier - 3	tier - 3	R.1012	22.24	4.29	yes	No	No	No major surgery	No
Id2329	1993	Jun	1	0	650	tier - 3	tier - 3	R.1013	17.07	5.22	No	No	Yes	1	No
Id2328	1995	Jul	4	0	650	tier - 3	tier - 3	R.1013	17.82	5.26	yes	No	No	1	No
Id2327	2002	Nov	29	0	668	tier - 3	tier - 2	R.1012	21.77	10.67	No	No	No	No major surgery	No
Id2326	1997	Nov	9	0	670	tier - 3	tier - 3	R.1013	20.1	5.6	yes	No	Yes	1	No
Id2325	2001	Sep	12	0	687.54	tier - 3	tier - 2	R.1013	24.76	4.54	yes	No	No	No major surgery	No

The output pane shows the execution of the SQL script. The first statement, 'CREATE TABLE merged\_data AS SELECT H.\*', affected 2333 records. The second statement, 'SELECT \* FROM merged\_data LIMIT 0, 1000', returned 1000 rows.

STEP 2: Add a Primary Key constraint for these columns

CODE:

ALTER TABLE merged\_data

ADD PRIMARY KEY (`Customer ID`(255));

DESC merged\_data;

OUTPUT:

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'health\_insurance' schema. The central pane shows the SQL script for adding a primary key constraint to the 'merged\_data' table. The right pane displays the 'Result Grid' with 10 columns: Field, Type, Null, Key, Default, and Extra. The data is limited to 1000 rows.

Field	Type	Null	Key	Default	Extra
Customer ID	text	NO	PK1		
year	int	YES			
month	text	YES			
date	int	YES			
children	int	YES			
charges	double	YES			
Hospital tier	text	YES			
City tier	text	YES			
State ID	text	YES			
BMI	double	YES			
HBA1C	double	YES			
Heart Issues	text	YES			
Any Transplants	text	YES			

The output pane shows the execution of the SQL script. The first statement, 'CREATE TABLE merged\_data AS SELECT H.\*', affected 2333 records. The second statement, 'SELECT \* FROM merged\_data LIMIT 0, 1000', returned 1000 rows. The third statement, 'ALTER TABLE merged\_data ADD PRIMARY KEY (Customer ID(255))', affected 0 rows. The fourth statement, 'DESC merged\_data', returned 16 rows.

STEP 3: clear null and missing value from merged data

CODE:

```
CREATE TABLE merged_data_cleaned AS

SELECT *

FROM merged_data

WHERE `Customer ID` IS NOT NULL

AND `Customer ID` <> ''

AND `year` IS NOT NULL

AND `month` IS NOT NULL

AND `date` IS NOT NULL

AND `charges` IS NOT NULL

AND `Hospital tier` IS NOT NULL

AND `City tier` IS NOT NULL

AND `State ID` IS NOT NULL

AND BMI IS NOT NULL

AND HBA1C IS NOT NULL

AND `Heart Issues` IS NOT NULL

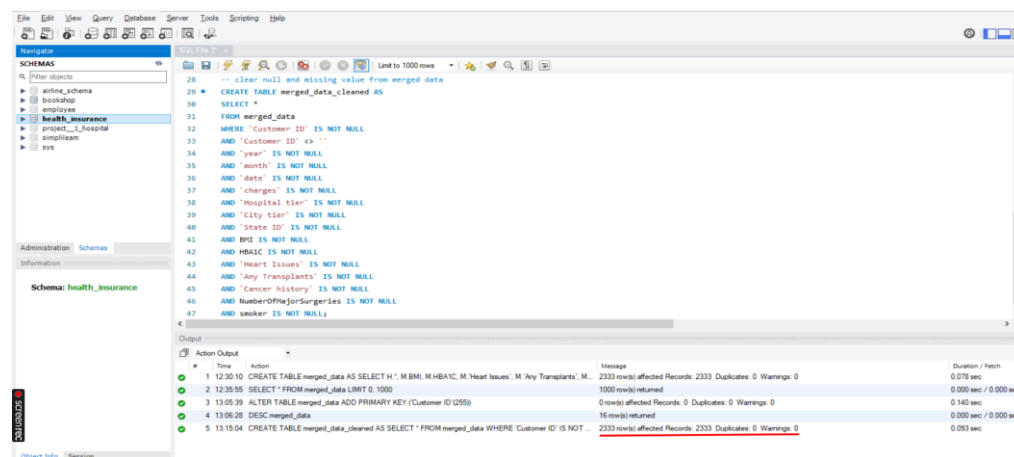
AND `Any Transplants` IS NOT NULL

AND `Cancer history` IS NOT NULL

AND NumberOfMajorSurgeries IS NOT NULL

AND smoker IS NOT NULL;
```

OUTPUT:



CODE:

SELECT \* FROM merged\_data\_cleaned;

OUTPUT:

Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID	BMI	HBA 1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker
Id1	1968	Oct	12	0	63770.43	tier - 1	tier - 3	R1013	47.41	7.47	No	No	No	No major surgery	yes
Id10	1978	Dec	29	0	48885.14	tier - 1	tier - 2	R1013	38.06	10.79	No	No	No	No major surgery	yes
Id100	1977	Jun	27	2	40284.38	tier - 1	tier - 3	R1012	48.2	4.84	No	No	No	No major surgery	yes
Id1000	1989	Dec	17	3	11250.43	tier - 3	tier - 2	R1026	39.17	4.15	No	No	No	No major surgery	No
Id1001	1969	Dec	30	2	11244.38	tier - 3	tier - 1	R1016	26.41	5.99	yes	No	Yes	1	No
Id1002	1976	Jun	28	2	11217.35	tier - 3	tier - 2	R1025	30.63	5.8	yes	No	No	No major surgery	No
Id1003	1970	Jun	14	2	11187.66	tier - 3	tier - 2	R1012	31.73	7.32	yes	No	No	2	No
Id1004	1972	Sep	3	0	11186.2	tier - 3	tier - 2	R1021	30.7	5.16	No	No	No	2	No
Id1005	1966	Aug	6	0	11165.42	tier - 3	tier - 1	R1016	25.935	5.96	yes	No	No	2	No
Id1006	1969	Jun	25	2	11163.57	tier - 3	tier - 2	R1011	35.9	4.85	yes	No	Yes	1	No
Id1007	1969	Nov	30	2	11150.78	tier - 3	tier - 2	R1011	26.7	5.09	yes	No	Yes	1	No
Id1008	1980	Aug	20	2	11103.33	tier - 3	tier - 1	R1021	33.71	4.94	No	No	No	No major surgery	No
Id1009	1966	Jul	5	0	11093.62	tier - 3	tier - 1	R1013	41.91	4.92	yes	No	No	2	No
Id101	1981	Oct	4	1	40273.65	tier - 1	tier - 3	R1013	35.75	8	yes	No	No	No major surgery	yes
Id1010	1966	Sep	9	0	11090.72	tier - 3	tier - 1	R1013	39.82	6.05	yes	No	No	2	No
Id1011	1972	Oct	7	3	11085.59	tier - 3	tier - 2	R1012	28.12	4.67	No	No	No	2	No
Id1012	1967	Sep	4	0	11082.58	tier - 3	tier - 2	R1012	26.98	9.81	yes	No	No	No major surgery	No
Id1013	1966	Nov	20	0	11073.18	tier - 3	tier - 3	R1011	27.2	6.06	yes	No	No	2	No
Id1014	1966	Nov	7	0	11070.54	tier - 3	tier - 3	R1011	25.3	5.19	yes	No	No	2	No
Id1015	1971	Nov	9	0	11068.77	tier - 3	tier - 2	R1012	30.25	10.16	No	No	No	No major surgery	No

STEP 4:

CODE:

ALTER TABLE merged\_data\_cleaned

ADD COLUMN Age INT;

SET SQL\_SAFE\_UPDATES = 0;

UPDATE merged\_data\_cleaned

SET Age = YEAR(CURRENT\_DATE) - year -

CASE

WHEN MONTH(CURRENT\_DATE) < month

OR (MONTH(CURRENT\_DATE) = month AND DAY(CURRENT\_DATE) < date) THEN 1

ELSE 0

END;

SET SQL\_SAFE\_UPDATES = 1;

SELECT \* FROM merged\_data\_cleaned;



OUTPUT:

SQL Script:

```

55 ALTER TABLE merged_data_cleaned
56 ADD COLUMN Age INT;
57 SET SQL_SAFE_UPDATES = 0;
58 UPDATE merged_data_cleaned
59 SET Age = YEAR(CURRENT_DATE) - year -
60 CASE
61 WHEN MONTH(CURRENT_DATE) < month
62 OR (MONTH(CURRENT_DATE) = month AND DAY(CURRENT_DATE) < date) THEN 1
63 ELSE 0
64 END;
65 SET SQL_SAFE_UPDATES = 1;
66 SELECT * FROM merged_data_cleaned;

```

Results Grid:

Customer ID	year	month	date	children	charges	hospital tier	city tier	state ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	Number Of Major Surgeries	smoker	Age
Id1	1968	Oct	12	0	63770.43	tier - 1	tier - 3	R1013	47.41	7.47	No	No	No	No major surgery	yes	56
Id10	1978	Dec	29	0	48885.14	tier - 1	tier - 2	R1013	38.06	10.79	No	No	No	No major surgery	yes	46
Id100	1977	Jun	27	2	40284.30	tier - 1	tier - 3	R1012	48.2	4.84	No	No	No	No major surgery	yes	47
Id1000	1989	Dec	17	3	11250.43	tier - 3	tier - 2	R1026	39.17	4.15	No	No	No	No major surgery	No	35
Id1001	1969	Dec	30	2	11244.38	tier - 3	tier - 1	R1016	26.41	5.99	yes	No	Yes	1	No	55
Id1002	1976	Jun	28	2	11217.35	tier - 3	tier - 2	R1025	30.63	5.8	yes	No	No	No major surgery	No	48
merged_data_cleaned 13	1978	Jun	14	7	11187.66	tier - 1	tier - 2	R1017	11.71	7.17	yes	No	No	7	No	64

Action Output:

#	Time	Action	Message	Duration / Fetch
3	13:44:45	ALTER TABLE merged_data_cleaned ADD COLUMN Age INT	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
4	13:45:11	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
5	13:46:10	UPDATE merged_data_cleaned SET Age = YEAR(CURRENT_DATE) - year - CASE WHEN MONTH(CURRENT_DATE) < month OR (MONTH(CURRENT_DATE) = month AND DAY(CURRENT_DATE) < date) THEN 1 ELSE 0 END	2333 row(s) affected Rows matched: 2333 Changed: 2333 Warnings: 0	0.094 sec
6	13:46:31	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
7	13:46:55	SELECT * FROM merged_data_cleaned LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.016 sec

TASK 2: Retrieve information about people who are diabetic and have heart problems with their average age, the average number of dependent children, average BMI, and average hospitalization costs

CODE:

```

SELECT AVG(Age) AS 'Average Age',
       AVG(Children) AS 'Average Children',
       AVG(BMI) AS 'Average BMI',
       AVG(Charges) AS 'Average Charges'
FROM merged_data_cleaned
WHERE HBA1C > 6 AND `Heart Issues` = 0;

```

OUTPUT:

SQL Script:

```

68 -- TASK 2
69 -- Retrieve information about people who are diabetic and have heart problems with their average age, the average number of dependent children, average BMI, and average hospitalization costs
70 SELECT AVG(Age) AS 'Average Age',
71        AVG(Children) AS 'Average Children',
72        AVG(BMI) AS 'Average BMI',
73        AVG(Charges) AS 'Average Charges'
74 FROM merged_data_cleaned
75 WHERE HBA1C > 6 AND `Heart Issues` = 0;

```

Results Grid:

Average Age	Average Children	Average BMI	Average Charges
47.2658	0.9283	30.932779005736178	15289.9298374761

Action Output:

#	Time	Action	Message	Duration / Fetch
1	14:23:10	SELECT AVG(Age) AS 'Average Age', AVG(Children) AS 'Average Children', AVG(BMI) AS 'Average BMI', AVG(Charges) AS 'Average Charges' FROM merged_data_cleaned WHERE HBA1C > 6 AND `Heart Issues` = 0	1 row(s) returned	0.000 sec / 0.000 sec



TASK 3: The average hospitalization cost for each hospital tier and each city level

CODE:

```
SELECT `Hospital tier`, `City tier`, AVG(Charges) AS 'Average Charges'
```

```
FROM merged_data_cleaned
```

```
GROUP BY `Hospital tier`, `City tier`;
```

OUTPUT:

```
75 WHERE HBAIC > 6 AND 'Heart Issues' = 0j
76
77
78 -- TASK 3
79 -- The average hospitalization cost for each hospital tier and each city level
80 SELECT `Hospital tier`, `City tier`, AVG(Charges) AS 'Average Charges'
81 FROM merged_data_cleaned
82 GROUP BY `Hospital tier`, `City tier`;
```

Hospital tier	City tier	Average Charges
ter -1	ter -3	33893.925675675706
ter -1	ter -2	28788.457476635507
ter -3	ter -2	9263.42747747477
ter -3	ter -1	9775.389793389431
ter -3	ter -3	9342.179912280704
ter -1	ter -1	29519.600813953486
ter -2	ter -3	12161.222811037523
ter -2	ter -1	11515.412628039694
ter -2	ter -2	11973.65534467636
ter -3	?	770.38
?	ter -3	700

Result 16

Output

Action Output

1 16:47:19 SELECT `Hospital tier`, `City tier`, AVG(Charges) AS 'Average Charges' FROM merged\_data\_cleaned GROU... 11 row(s) returned 0.000 sec / 0.000 sec

TASK 4: The number of people who have had major surgery with a history of cancer

CODE:

```
SELECT COUNT(*) AS 'No:of People'
```

```
FROM merged_data_cleaned
```

```
WHERE NumberOfMajorSurgeries = 1 AND `Cancer history` = 'Yes';
```

OUTPUT:

```
82 GROUP BY `Hospital tier`, `City tier`;
```

```
83
84
85 -- TASK 4
86 -- the number of people who have had major surgery with a history of cancer
87 SELECT COUNT(*) AS 'No:of People'
88 FROM merged_data_cleaned
89 WHERE NumberOfMajorSurgeries = 1 AND `Cancer history` = 'Yes';
```

No:of People
791

Result 17

Output

Action Output

1 16:52:27 SELECT COUNT(\*) AS 'No:of People' FROM merged\_data\_cleaned WHERE NumberOfMajorSurgeries = 1 AND `Cancer history` = 'Yes'; 1 row(s) returned 0.000 sec / 0.000 sec

TASK 5: The number of tier-1 hospitals in each state

CODE:

```
SELECT `State ID`, COUNT(*) AS 'No:of Tier1 Hospitals'
FROM merged_data_cleaned
WHERE `Hospital tier` = 'tier - 1'
GROUP BY `State ID`;
```

OUTPUT:

The screenshot displays a SQL IDE interface. On the left, a 'SCHEMAS' pane shows a tree of databases including 'airline\_schema', 'bookshop', 'employee', 'health\_insurance', 'project\_1\_hospital', 'simpleteam', and 'sys'. The main editor window contains the following SQL code:

```
-- TASK 5
-- the number of tier-1 hospitals in each state
92 SELECT `State ID`, COUNT(*) AS 'No:of Tier1 Hospitals'
93 FROM merged_data_cleaned
94 WHERE `Hospital tier` = 'tier - 1'
95 GROUP BY `State ID`;
```

Below the code, the 'Result Grid' shows the output of the query. It is a table with two columns: 'State ID' and 'No:of Tier 1 Hospitals'. The data is as follows:

State ID	No:of Tier 1 Hospitals
R1013	67
R1012	63
R1011	116
R1014	10
R1017	7
R1015	2
R1016	8
R1024	14
R1019	5
7	2
R1018	1
R1026	5
R1023	4

At the bottom, the 'Output' pane shows the execution log. It includes a message: '1 16:58:18 SELECT `State ID`, COUNT(\*) AS `No of Tier1 Hospitals` FROM merged\_data\_cleaned WHERE `Hospital tier` = 'tier - 1' GROUP BY `State ID`; 13 row(s) returned'. The duration of the query is listed as '0.016 sec / 0.000 sec'.