**Operations performed writeup:**

1. Create a database “airlines” and use it for the rest of the project. (SS1)
2. Import the CSV file “Customer” in the same database using the following steps:
3. Right-click on the tables option under the “airlines” database and select “Table Data Import Wizard”.
4. Browse the Customer.csv file from the dialogue box. Then click Next.
5. Select Create new table from the Select Destination dialogue box and click on Next.
6. Then configure the import settings like the column names and data type. Click Next.
7. Import Data dialogue box will appear. Click on next to start the import and again click Next.
8. Data will be imported and then finish. (SS2 to SS8)
9. Repeat step 2 for the rest of the tables (passengers\_on\_flights.csv, routes.csv and ticket\_deatils.csv). All the tables appear under the Tables option in the airlines database. (SS9)
10. Then create an ER Diagram for the airlines database.
11. From the menu bar click on Database-> Reverse Engineer. (SS10)
12. Reverse Engineer Database dialogue box will appear. Set the parameters to connect to a database and then click on Next. (SS11)
13. In the next dialogue box connection with the database is made and relevant information is fetched. Click Next. (SS12)
14. Select the airlines schema and it will be included. Click on Next. (SS13)
15. Click next after the objects from the selected schema are retrieved. (SS14)
16. Select “Import the object” option from the dialogue box and click on Execute. (SS15)
17. Reverse Engineering Progress dialogue box will appear. Click on next once the operations are completed successfully. (SS16)
18. Click on Finish. (SS17)
19. The tables are ready. (SS18)
20. Right-click on the table and select the edit <table> option. (SS19)
21. Then set the primary key and foreign key for each table and make the ER Diagram. (SS20- SS22)
22. Write a query to create a route\_details table using suitable data types for the fields, such as route\_id, flight\_num, origin\_airport, destination\_airport, aircraft\_id, and distance\_miles. Implement the check constraint for the flight number and unique constraint for the route\_id fields. Also, make sure that the distance miles field is greater than 0.
23. The table is created using create table command and all the constraints are applied.
24. The table information is shown in detail using desc table command. (SS23)
25. Write a query to display all the passengers (customers) who have travelled in routes 01 to 25. Take data from the passengers\_on\_flights table.
26. The select command is written.
27. The customer\_id and route\_id is fetched from the table.
28. The between operator is used with the where clause to apply the condition.
29. The query has fetched 26 rows. (SS24)
30. Write a query to identify the number of passengers and total revenue in business class from the ticket\_details table.
31. The select command is written.
32. The sum aggregation function is applied on no\_of\_tickets column to identify the number of passengers and is given the alias 'Number of Passengers'.
33. Another, sum aggregation function is applied on the no\_of\_tickets\*Price\_per\_ticket column to find the total revenue and is given the alias 'Total Revenue'.
34. A where clause is also used to check the condition of class\_id as Bussiness.
35. The Number of Passengers is 13 and the Total Revenue is 6034. (SS25)
36. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.
37. The Select command is written.
38. The trim function is used on the first\_name and last\_name columns to remove any leading or trailing spaces.
39. Then the coalesce function is used on the trimmed first\_name and last\_name columns to return ‘’ in case of first\_name or last\_name is NULL.
40. Lastly the concat function is used on the results returned by coalesce function and is given the alias ‘Full Name’.
41. The query has fetched total 50 rows. (SS26)
42. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket\_details tables.
43. The Select command is written.
44. The tables customer and ticket\_details are joined with common column customer\_id.
45. Customer\_id and the name of the customer is fetched.
46. Distinct is used with customer\_id column to get the details only once in case the customer has booked multiple tickets.
47. The first-name and the last\_name is concatenated and displayed as Full Name in the output.
48. In the where clause p\_date and no\_of\_tickets is checked not to be null to get the customers who has registered and booked the tickets.
49. The query has fetched total 33 rows. (SS27)
50. Write a query to identify the customer’s first name and last name based on their customer ID and brand (Emirates) from the ticket\_details table.
51. The Select command is written.
52. The tables customer and ticket\_details are joined with the common column customer\_id.
53. Customer\_id, first\_name and last\_name are fetched from the customer table.
54. Distinct is used with customer\_id column to get the details only once in case the customer has booked multiple tickets.
55. The where clause is used to check the brand as Emirates
56. The query has fetched total 14 rows. (SS28)
57. Write a query to identify the customers who have travelled by *Economy Plus* class using Group By and Having clause on the passengers\_on\_flights table.
58. The Select command is written.
59. The customer id and class\_id is fetched.
60. The group by clause is used on customer id and class\_id.
61. The condition of Economy Plus class\_ id is checked using having clause.
62. The query has fetched total 9 rows. (SS29)
63. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket\_details table.
64. The Select command is used.
65. The aggregation function sum is used on no\_of\_tickets \* Price\_per\_ticket and given the alias ‘Revenue’.
66. If clause is used to check the revenue>10000 and the appropriate message is displayed.
67. The revenue is 15369 and the Message displayed is “The revenue has crossed 10000”. (SS30)
68. Write a query to create and grant access to a new user to perform operations on a database.
69. To create a user, create user command is used and a new user “Surbhi” is created.
70. Show privileges command is used to find all the available privileges.
71. Grant command is used to grant alter, create, delete, drop, index, insert, select, update, trigger, alter routine, create routine, execute, create temporary tables privileges to the new user.
72. Then Flush privileges command is used to save the changes.
73. All the granted privileges can be checked using Show Grants command. (SS31)
74. Write a query to find the maximum ticket price for each class using window functions on the ticket\_details table.
75. The Select command is used.
76. Distinct class\_id is selected.
77. Window function max is used to find maximum ticket price which is partitioned by class\_id.
78. The query has displayed 4 different class\_id’s and their maximum ticket price. (SS32)
79. Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers\_on\_flights table.
80. The query is written with where clause and the Execution Plan is checked. (SS33)
81. To improve the speed and performance index is created on route\_id.
82. The same query is executed again with improved speed and performance as verified from Execution Plan. (SS34)
83. For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.
84. The query is written before and after creating index and the execution plan is checked. (SS33 and SS34)
85. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.
86. The Select command is used.
87. The customer\_id is fetched.
88. The aggregation function sum is used on no\_of\_tickets\*Price\_per\_ticket and is given alias Total Price.
89. The group by clause is used on customer\_id with rollup.
90. A total of 34 rows are returned and the last column shows the total price column as the sum of all the values in the same column which is 15369. (SS35)
91. Write a query to create a view with only business class customers along with the brand of airlines.
92. A view is created as per the above query.
93. The Select command is used to fetch the view results.
94. A total of 13 rows are returned. (SS36)
95. Write a query to create a stored procedure to get the details of all passengers flying between a range of routes defined in run time. Also, return an error message if the table doesn't exist.
96. A stored procedure is created with two arguments for the routes which will be defined at runtime.
97. Then using the query, the existence of the table is checked.
98. If else is used to print the error message if the table does not exist otherwise show the passenger details.
99. To get the passenger details the customer and passengers\_on\_flights table is joined.
100. The call command is used to call the procedure with two arguments (SS37 and SS38)
101. Write a query to create a stored procedure that extracts all the details from the routes table where the travelled distance is more than 2000 miles.
102. A stored procedure without any arguments is created.
103. The select command with where clause is used to fetch the details.
104. The calling is done which has returned 24 rows. (SS39)
105. Write a query to create a stored procedure that groups the distance travelled by each flight into three categories. The categories are, short distance travel (SDT) for >=0 AND <= 2000 miles, intermediate distance travel (IDT) for >2000 AND <=6500, and long-distance travel (LDT) for >6500.
106. A stored procedure without any argument is created.
107. With clause is used to create a temporary table which contains all the columns and the new column with different categories.
108. The categories are created using the if elseif else statement.
109. The temporary table is used to fetch the newly added categories and their count.(SS40 and SS41)
110. Write a query to extract ticket purchase date, customer ID, class ID and specify if the complimentary services are provided for the specific class using a stored function in stored procedure on the ticket\_details table.

Condition:

If the class is *Business* and *Economy Plus,* then complimentary services are given as *Yes,* else it is *No*

1. A stored function is created with one argument and return type which will take class\_id as argument and return YES or NO based on the class\_id as per the query.
2. The Select statement is used to fetch the records. (SS42 and SS43)
3. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.
4. A curser is created inside the procedure as per the question.
5. The procedure is then called to get the output. (SS44 and SS45).