Vidhi Pankaj Jadav	U 7447 0957	vjadav@mail.usf.edu
Pranali Mohan Kanade	U 7056 6891	pmk1@mail.usf.edu
Soumitra Korde	U 6946 4200	korde@mail.usf.edu
Rishab Agarwal	U 8983 9745	rishaba@mail.isf.edu

Airlines Management System Database Project

Advanced Database Management Systems

Prof. Don Berndt

Contents

Introduction

Airline management system database is a project planned to comprehend an operational database system for an airline. This project will help us understand the fundamentals of a relational database and the structured query language (SQL) used to design this project. It will be implemented on Oracle Sql Developer. Database is designed to analyse and implement Airlines and able to manage Flights, plane, booking and employee. Database is built and then defined constraint like Primary key and foreign key to eliminate redundancy of data. We have implemented performance optimization and built some queries to perform some interesting activity. We have used Tableau for Data Visualization purpose.

1. Database Design

We are representing flight booking and management system in the entity-relationship(ER) diagram. The ER diagram of Airline System consists of tables such as Flight, Passenger, Airport, Employee, Airline and Booking entities and their relation between each other. The details of every flight booking are stored in Flight and likewise for Passenger, Airport, Booking, Plane, Employee entities. Each entity contains primary key. Passenger, Airport, Booking, Plane, Employee entities are bound with Flight entity with foreign keys. There is one to one relation between Flight and Plane and all other relations are one to many.

1.1. Airline management system has below entities and attribute

1. **Flight:** Attribute of Flight entity are Flight_Number, Flight_Class, Flight_Source, Flight_Destination, Flight_Departure_Time, Alight_Arrival_Time, Airport_Code, Passenger_ID

2. **Passenger:** Attribute of Passenger entity are Passenger_Name, Passenger_Age, Passenger_Sex, Passenger_ID, Passenger_Address, Passenger_Email, Flight_No

3. **Airport:** Attributes of Airport entity are Airport_Code, City, Name, Rank_By_Popularity, Country

4. **Booking:** Attributes of Booking entity are Fare, Flight_Number, Booking_ID, Booking_Date, Travel_Agency

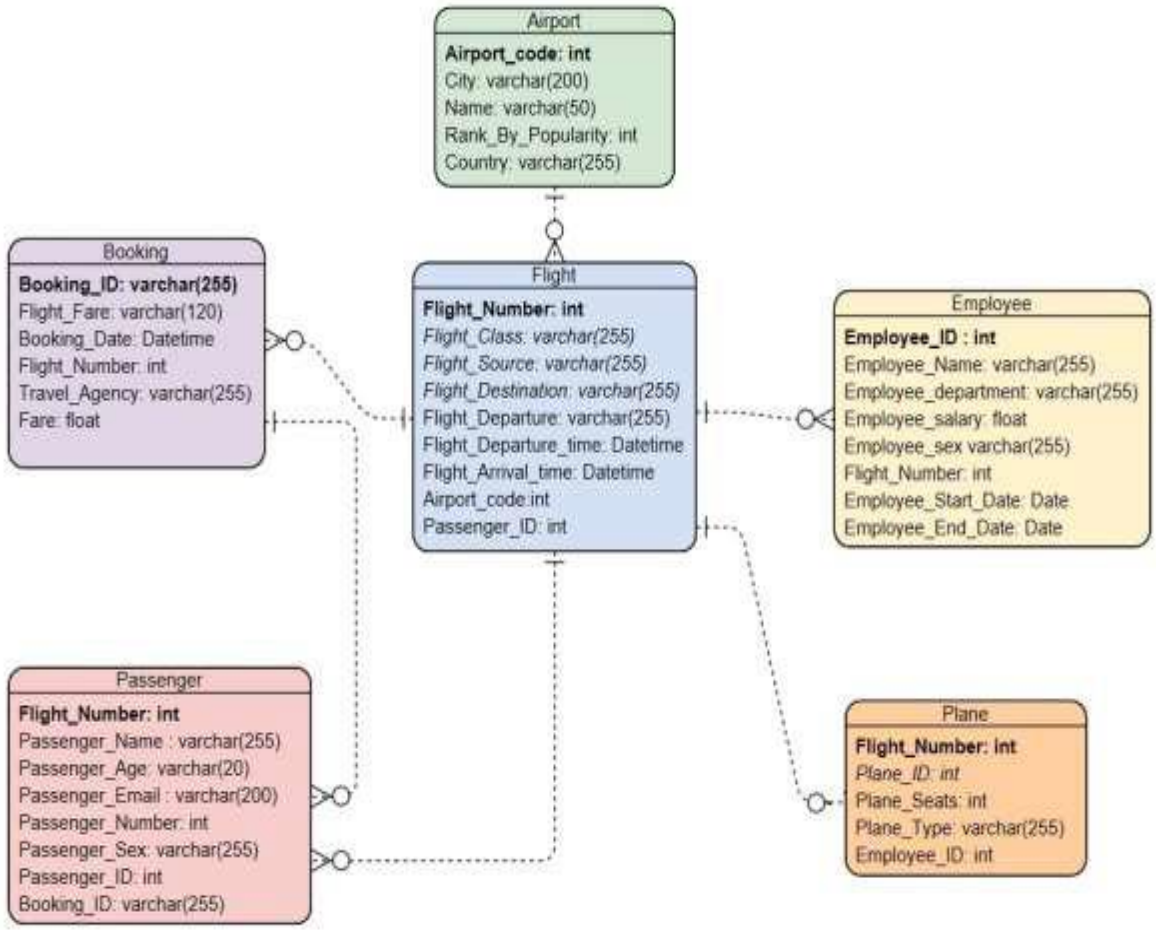
5. **Plane:** Attributes of Plane entity are Flight_Number, Plane_ID, Plane_Seats, Plane_Type,

Employee_ID

6. **Employee:** Attributes of Employee are Employee_ID, Employee_Name, Employee_Department,

Employee_Salary, Employee_Sex, Flight_Number, Employee_Start_date, Employee_End_Date

1.2. ER Diagram of Airlines Management System



2. Data Source

For generating the data, we have used Mockaroo realistic data generator website to generate random data and imported it to create the Airlines Management database which has tables and attributes. We have used fields type like Character sequence for random string generation, custom list to select class, city from source and destination and date. We have done data cleaning and manipulation after data is generated with help of website. CSV files is generated with the column names and fields are populated randomly as per our setting. Some fields need cleaning and manipulation which we have done using Excel. CSV files generated for every entity are then imported in Oracle.

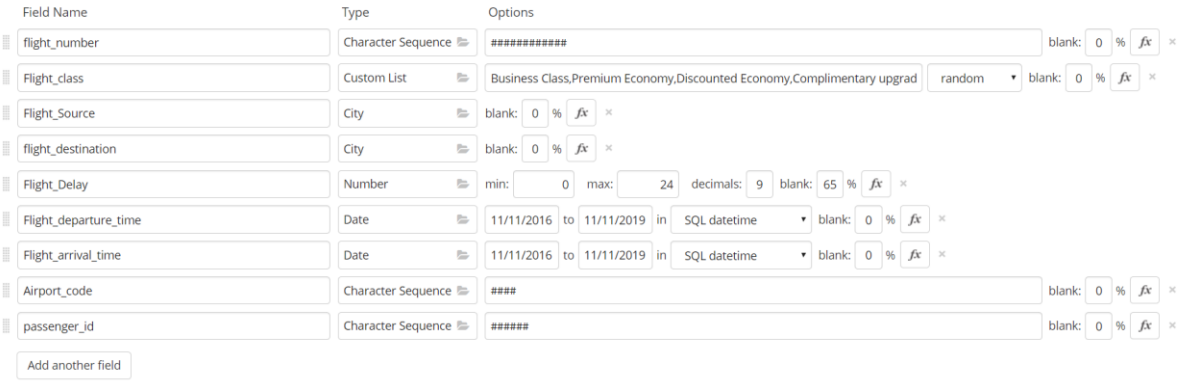
Link: https://mockaroo.com/

2.1. Flight data generation

Flight table has attributes named FLIGHT_NUMBER, FLIGHT_CLASS, FLIGHT_SOURCE, FLIGHT_DESTINATION, FLIGHT_DELAY, FLIGHT_DEPARTURE_TIME, FLIGHT_ARRIVAL_TIME, AIRPORT_CODE, and PASSENGER_NUMBER. With help of Mockaroo website we will be generating

random data. Departure time needed manipulation as it was randomly generated by website and records were having departure time which was not depicting real time scenario as some records were having arrival time before departure and some records were after taking years to reach destination. In order to generate data for flight_arrival_time which is more realistic we have used excel to generate data with help of function Randbetween() and date manipulation in excel.

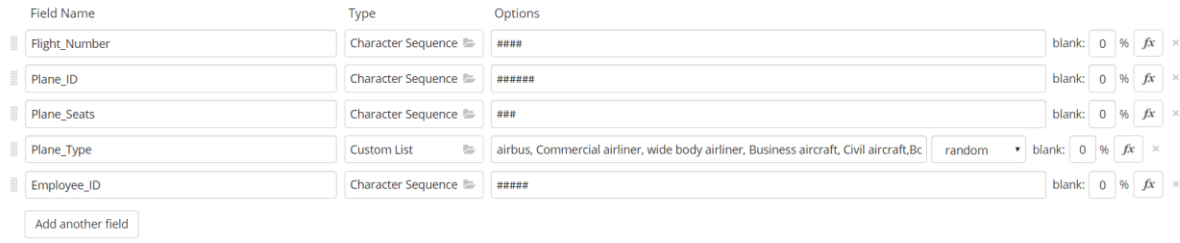
2.1.1. Flight data generation setting



2.2. Plane Data generation

Plane table has attributes like Flight_Number, Plane_ID, Plane_Seats, Plane_Type, Employee_ID. For generating Flight_Number, Plane_Id, Plane_Seats, Employee_ID records we have Character sequence and for Plane_Type we have used Custom list to select between options airbus, Commercial airliner, wide body airliner, Business aircraft, Civil aircraft, Boeing.

2.2.1. Plane data generation setting

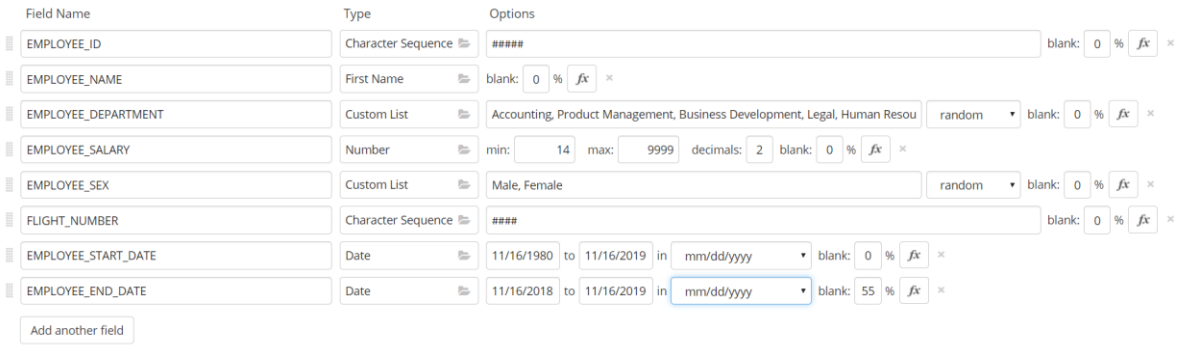


2.3. Employee Data generation

Employee has attributes named Employee_ID, Employee_Name, Employee_Department, Employee_Salary, Employee_Sex, Flight_Number, Employee_Start_date, Employee_End_Date. We have used character sequence for attributes Employee_ID, Flight_Number. For Employee_Name we have used First name generator. Employee_Department is populated with help of Custom list having record values as Accounting, Product Management, Business Development, Legal, Human Resources, Engineering, Services, Sales, Marketing, Training, Support, Research and Development.

Employee salary is generated with help of Number raning from 14 to 9999 having decimal digits as 2. Employee_Sex is generated with the help of Custom List including gender Male and Female. Employee_Start_Date and Employee_End_date is generated using Date. Employee_End_date has blank values for current employee and Date for Employee who has left is generated in Excel using Randbeetween() function as Website has Employe_End_Date even before Employee_Start_date for some records as data is generated randomly.

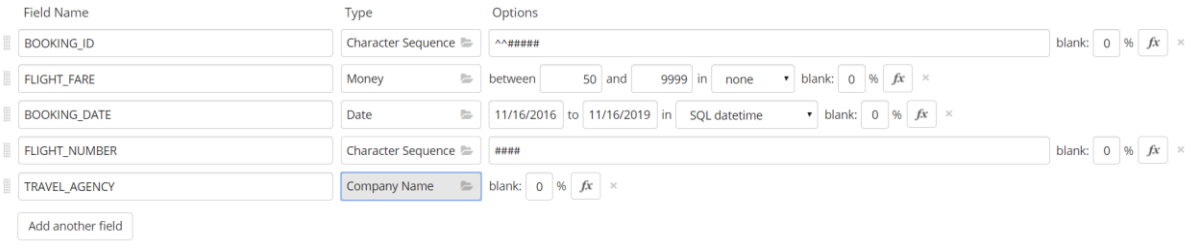
2.3.1. Employee data generation setting



2.4. Booking Data generation

Booking table has attributes named Booking_ID, Flight_Fare, Flight_Number, Booking_Date, Travel_Agency. Booking_ID and Flight_Number are generated using Character Sequence. Flight_Fare is generated using Money ranging from 50 to 9999. Booking_Date is generated from date and Travel_Agency from Company Name.

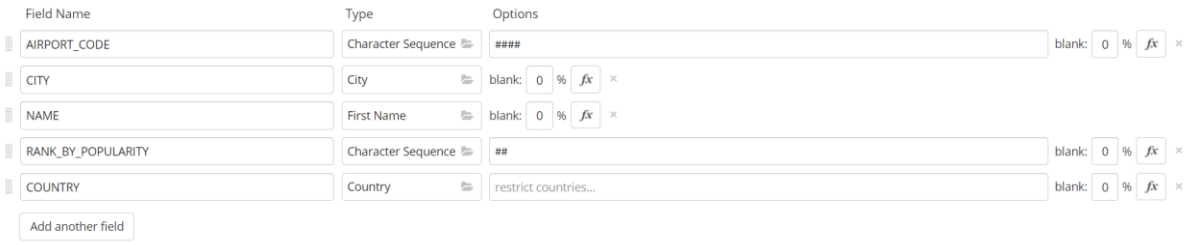
2.4.1. Booking data generation setting:



2.5. Airport Data generation

Airport has attributes named Airport_Code, City, Name, Rank_By_Popularity, Country. Airport_Code and Rank_By_Popularity is generated with Character Sequence Type. City is populated with City and Name with First Name. Country is generated with Country.

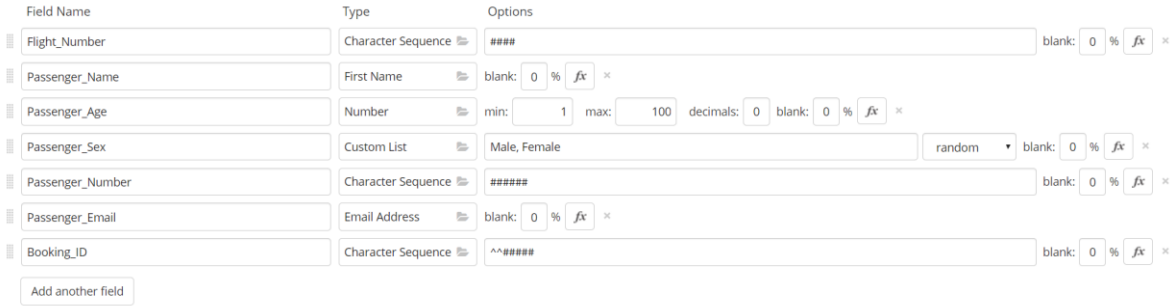
2.5.1. Airport data generation setting



2.6. Passenger Data generation

Passenger has attributes named Passenger_Name, Passenger_Age, Passenger_Sex, Passenger_Number, Passenger_Email, Flight_Number, Booking_ID. Flight_Number, Passenger_Number and Booking_Id is generated using Character sequence. Passenger_Name with First Name. Passenger_Name is generated using First Name and Passenger_Sex is populated using Custom List having Male and Female options. Passenger_Email is generated using Email address.

2.6.1 Passenger data generation setting

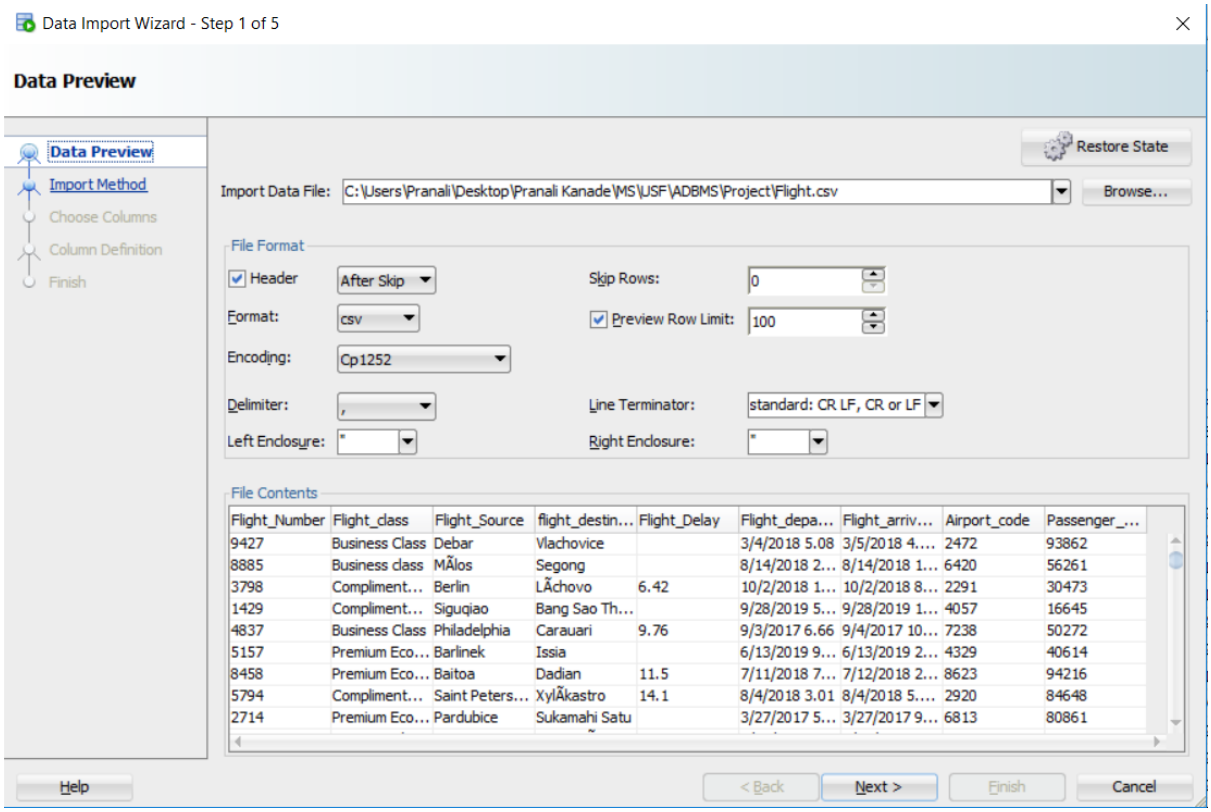


3. Data import in Oracle SQL Developer

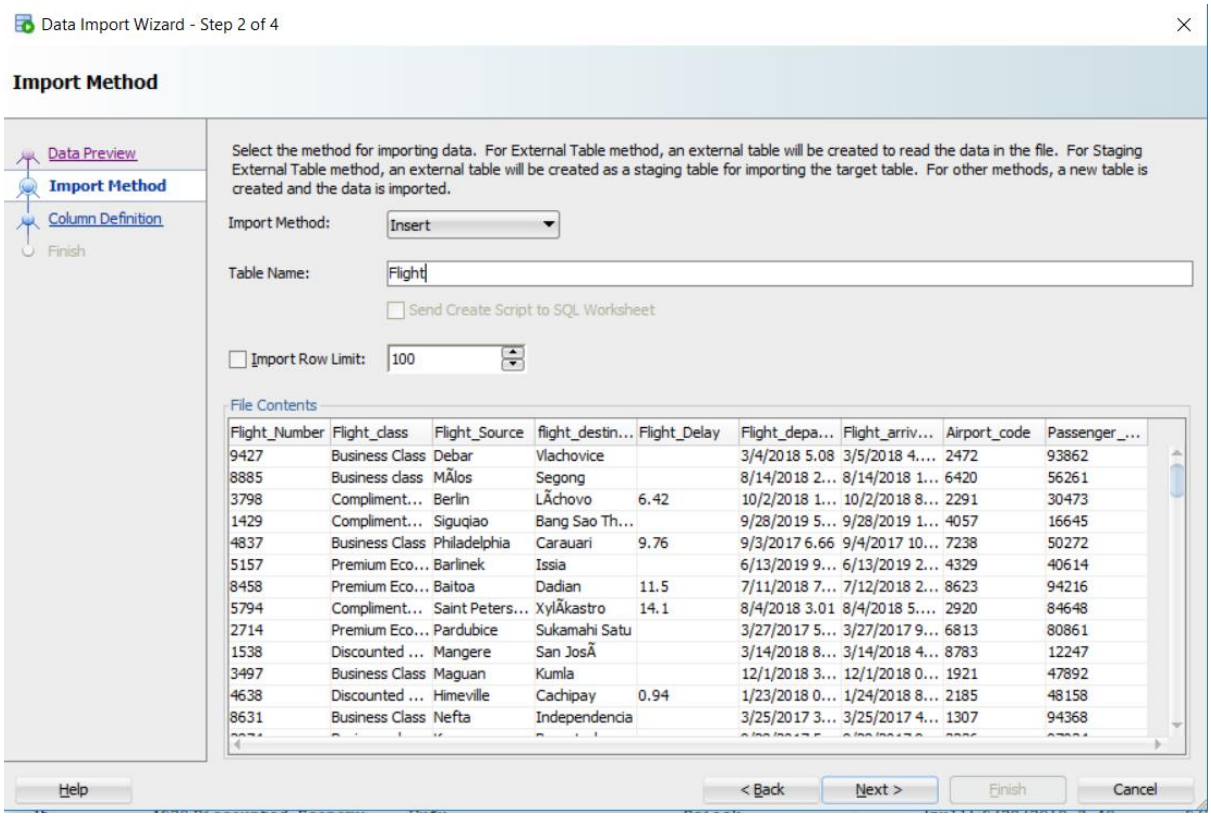
Entities viz Flight, Plane, Passenger, Employee, Booking, Airport CSV files are generated using website are now ready for import after above setting and some cleaning and manipulation done using Excel. Import of every Entity CSV is done using Import functionality of Oracle SQL Database. Following steps are followed to import every entity file.

3.1. Import Flight.csv file

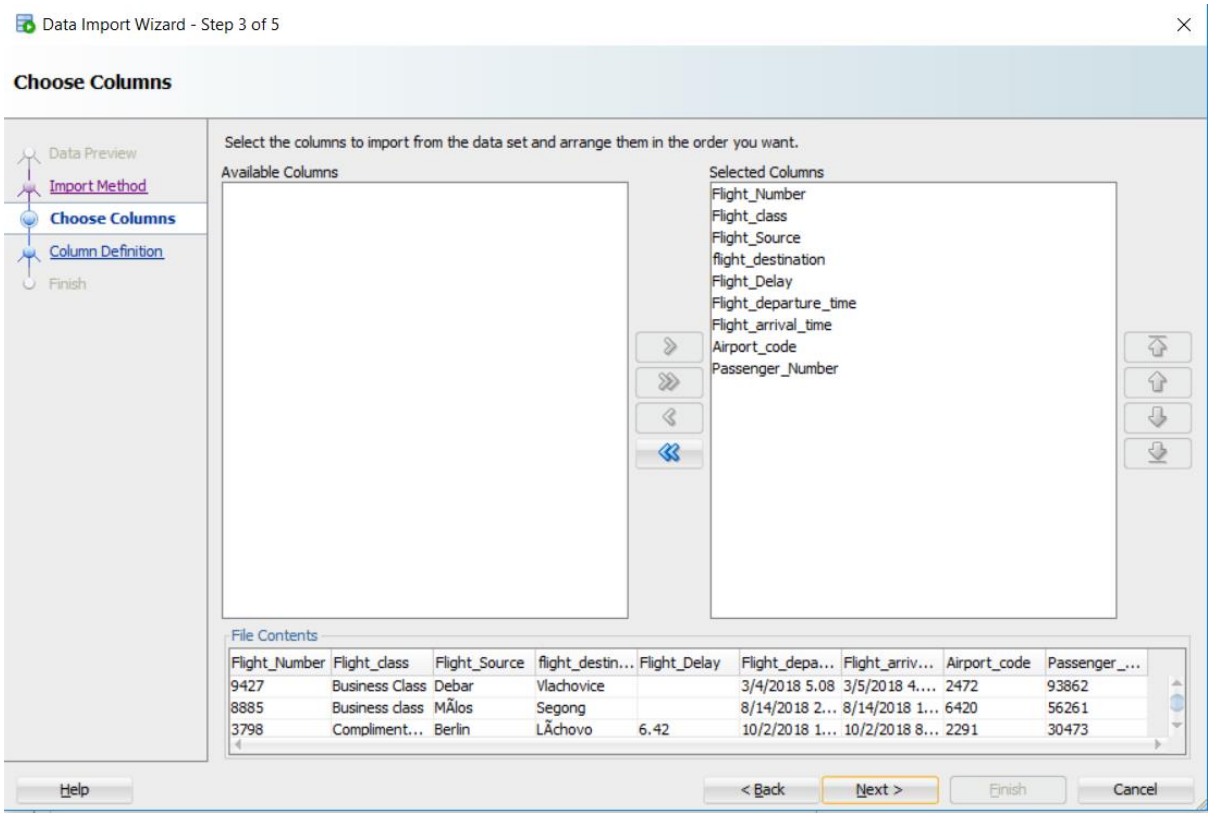
a) Flight.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



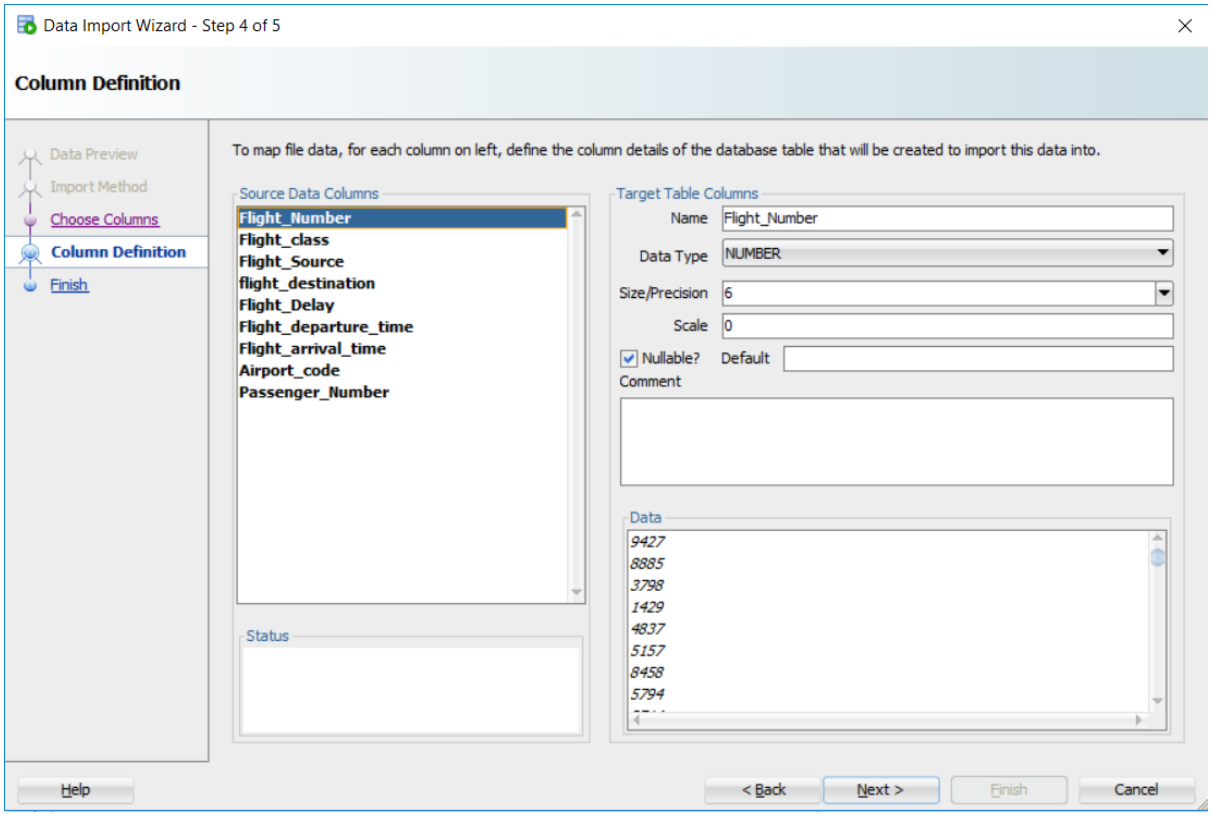
b) Import Method is selected as Insert and provided Table Name as Flight.



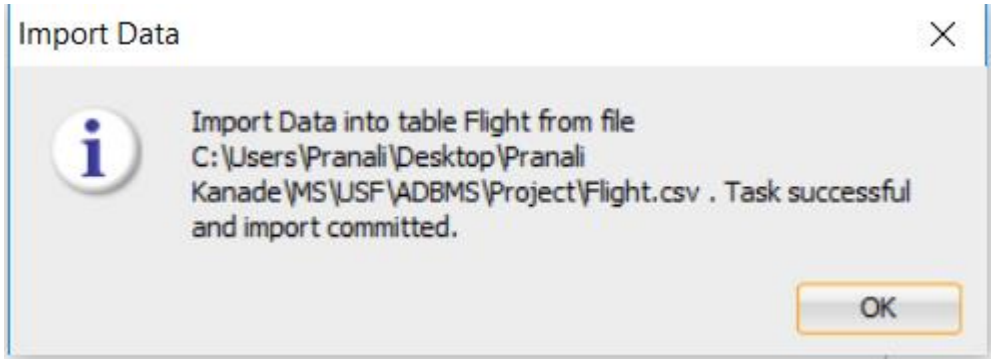
c) Columns are selected in Choose Columns tab as below.



d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

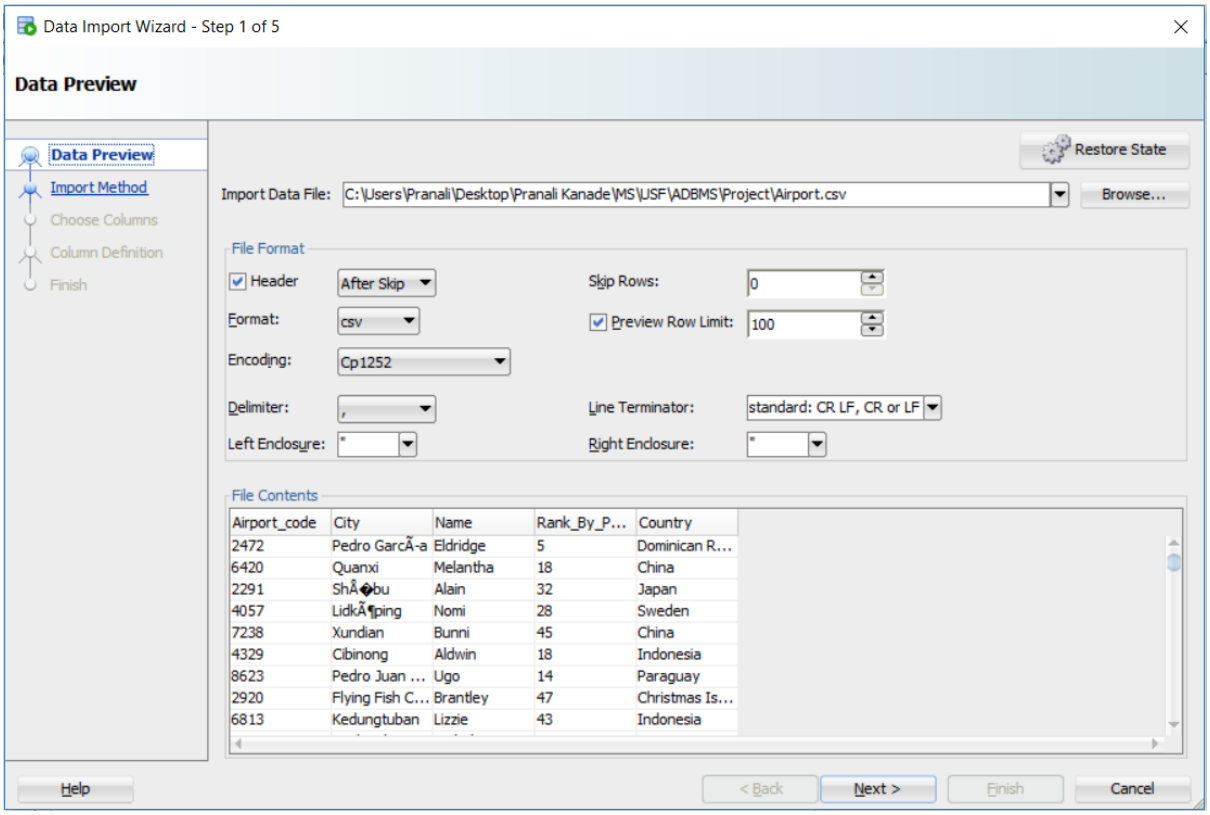


e) After clicking Next and Submit we can see below pop up where we can see Table is imported successfully using CSV file.

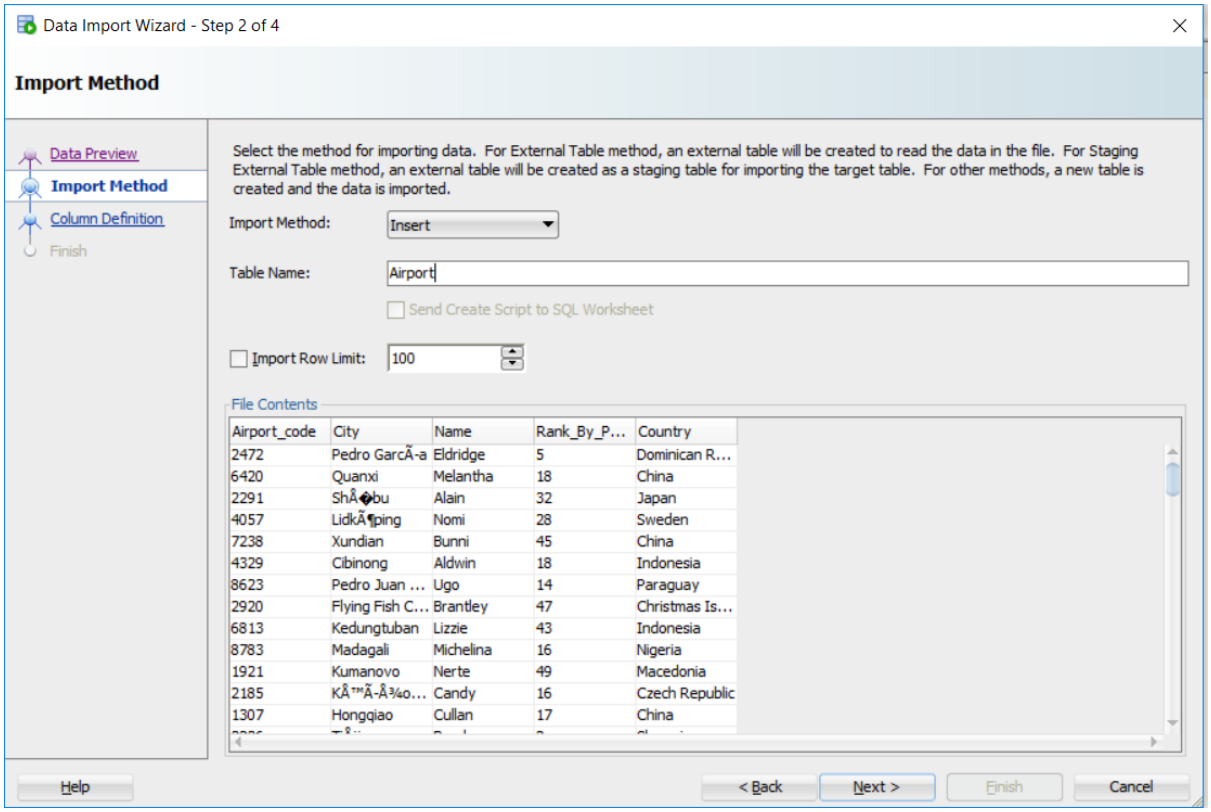


3.2. Import Airport.csv file

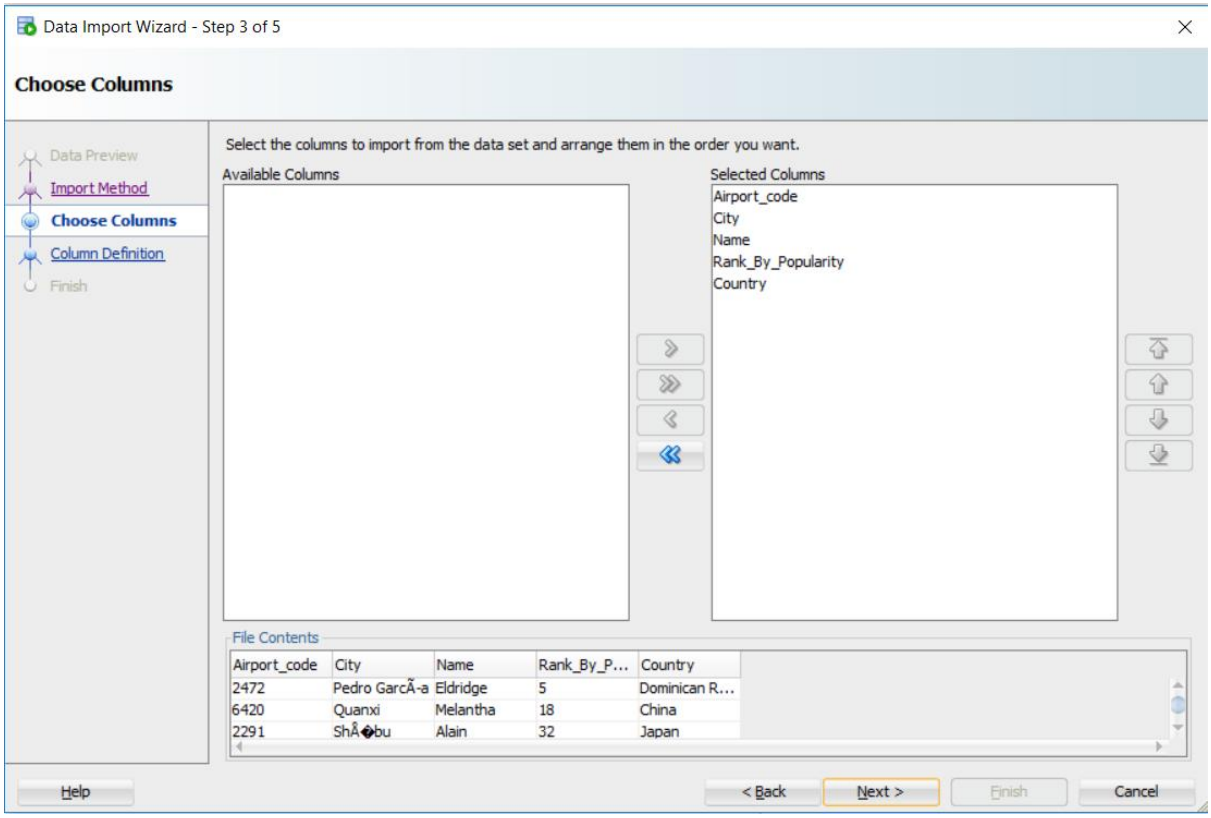
a) Airport.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



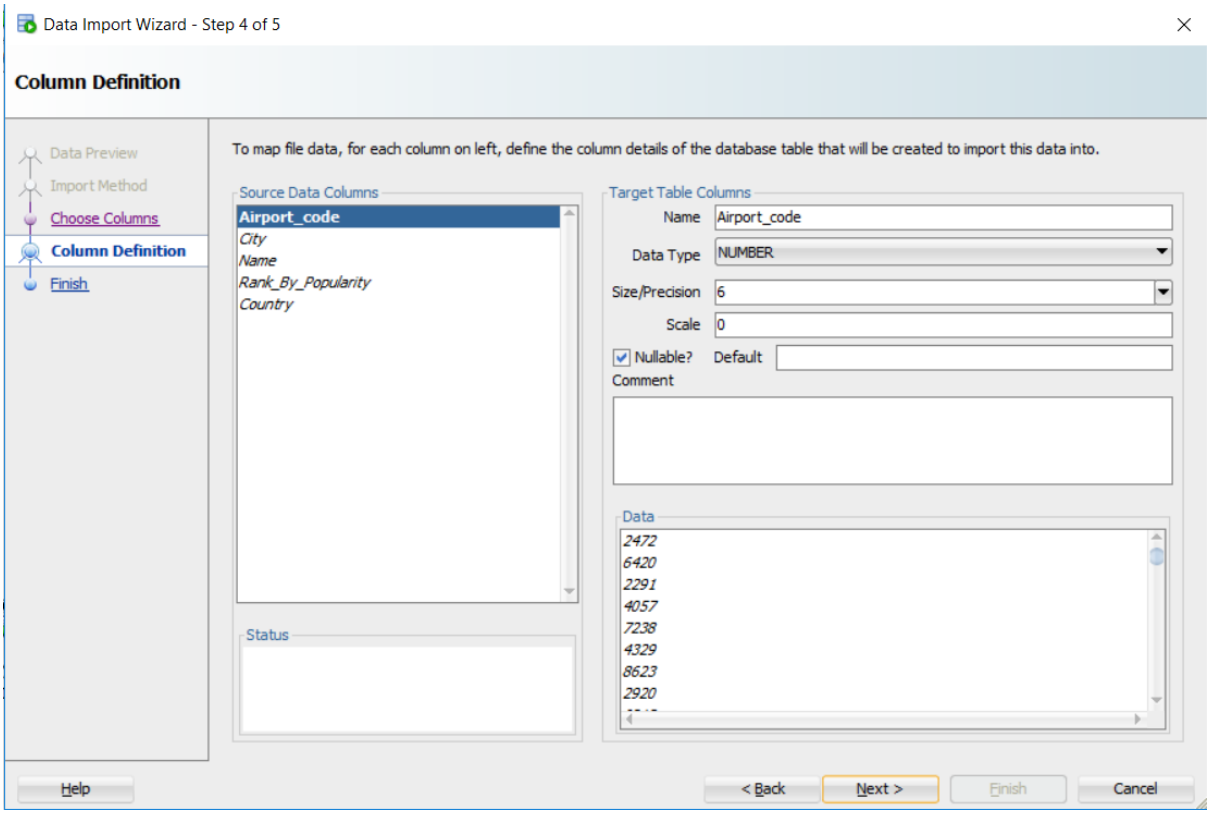
b) Import Method is selected as Insert and provided Table Name as Airport.



c) Columns are selected in Choose Columns tab as below.

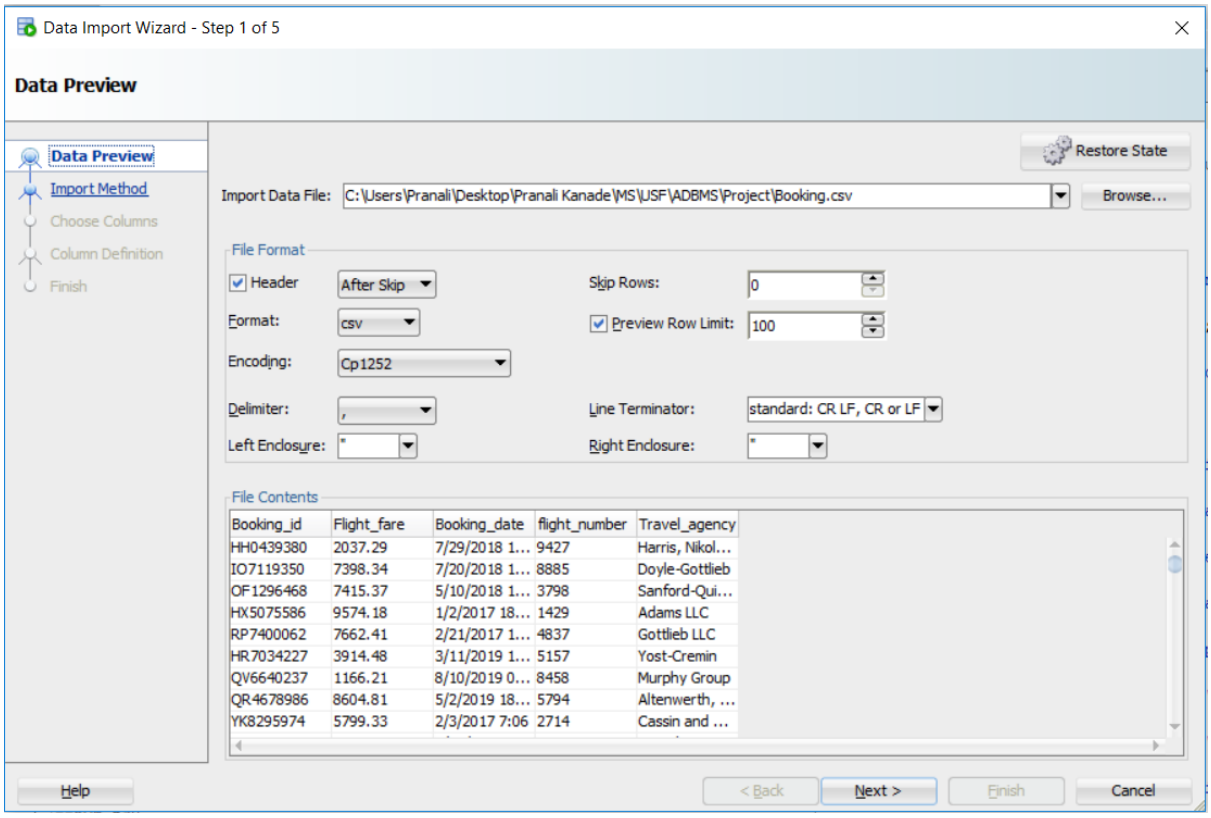


d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

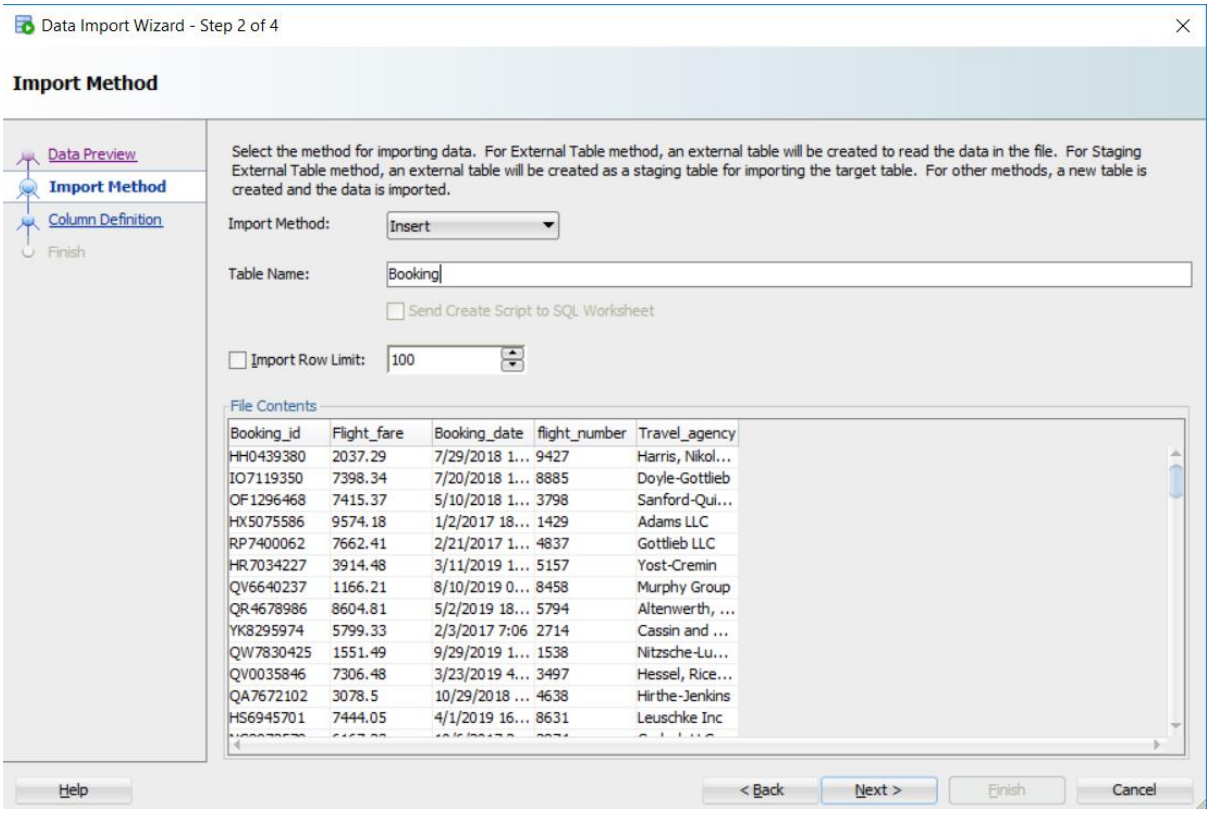


3.3. Import Booking.csv file

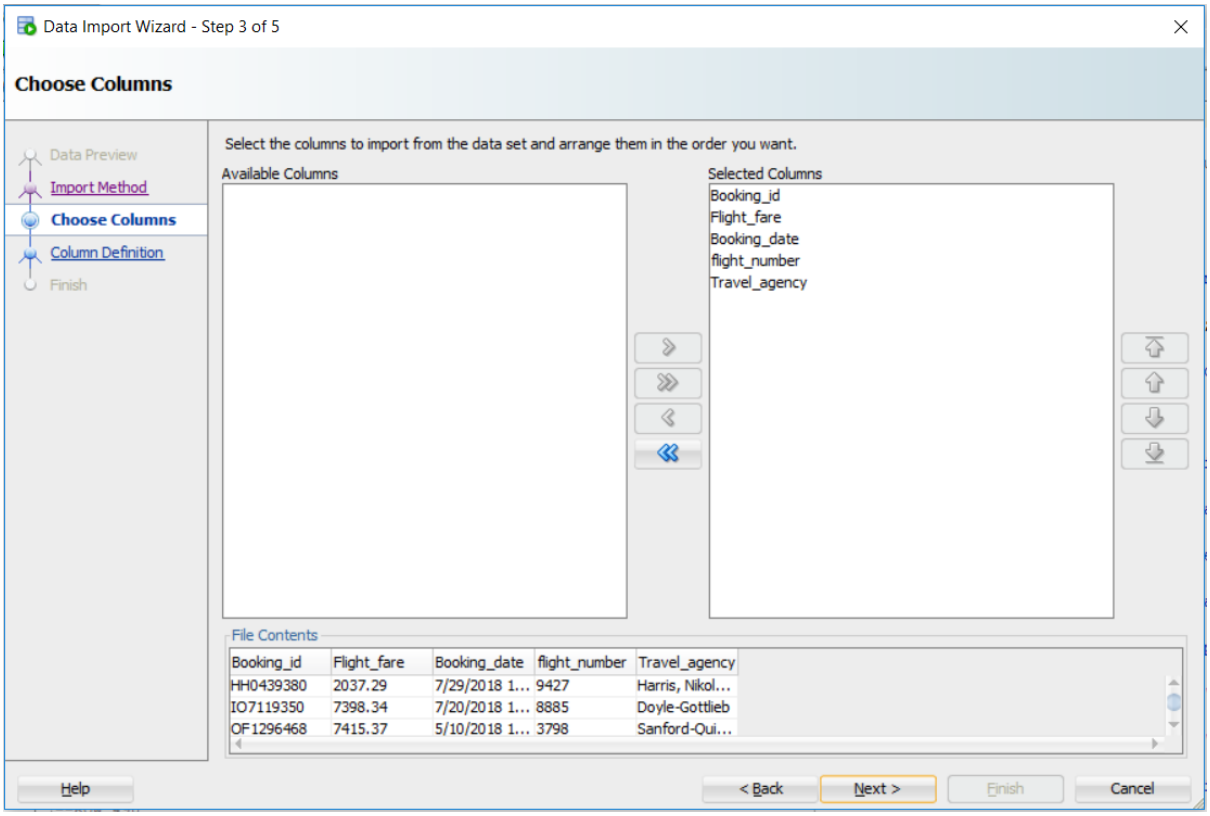
a) Booking.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



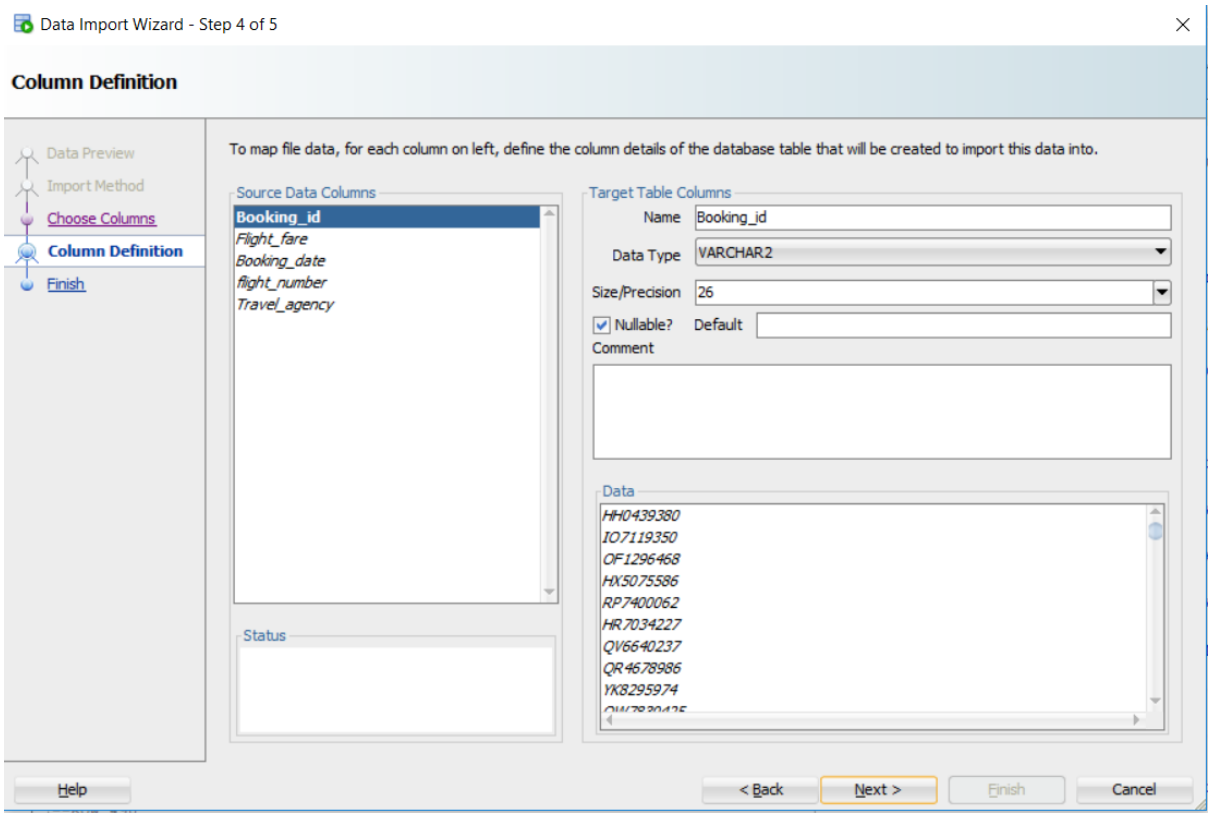
b) Import Method is selected as Insert and provided Table Name as Booking.



c) Columns are selected in Choose Columns tab as below.

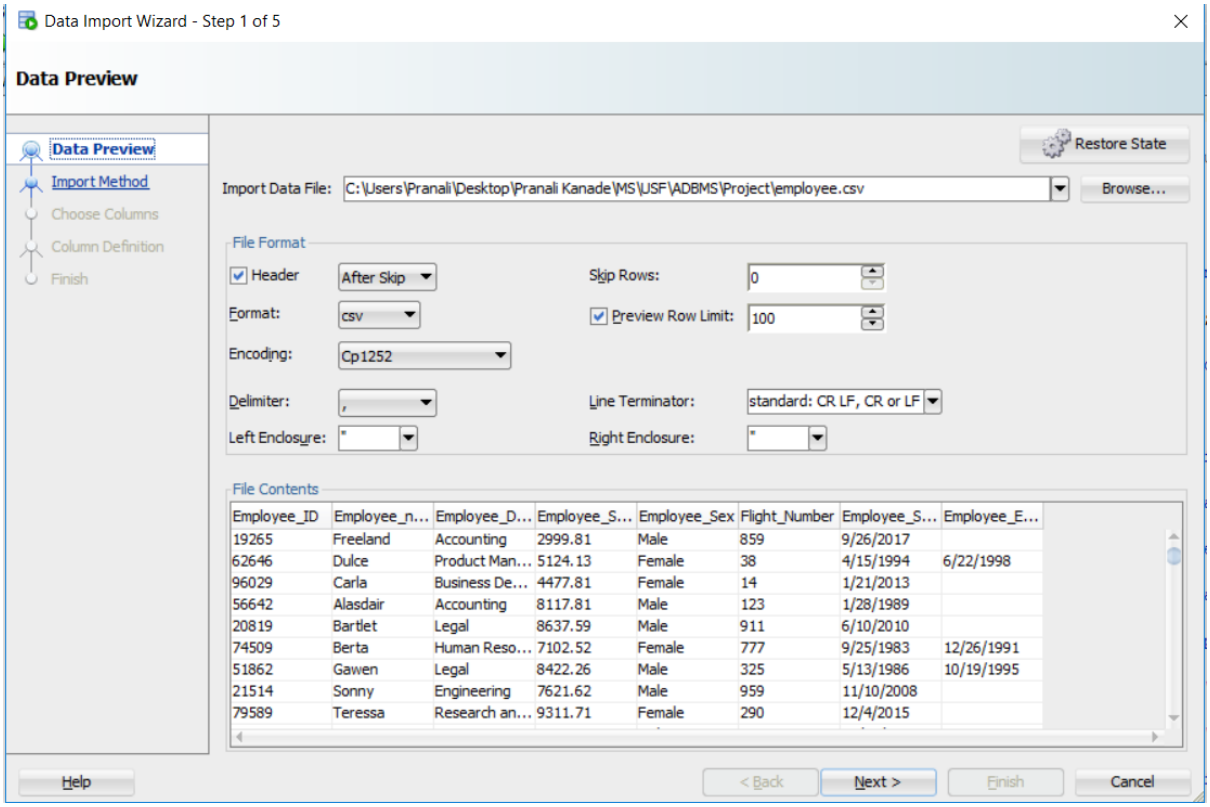


d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

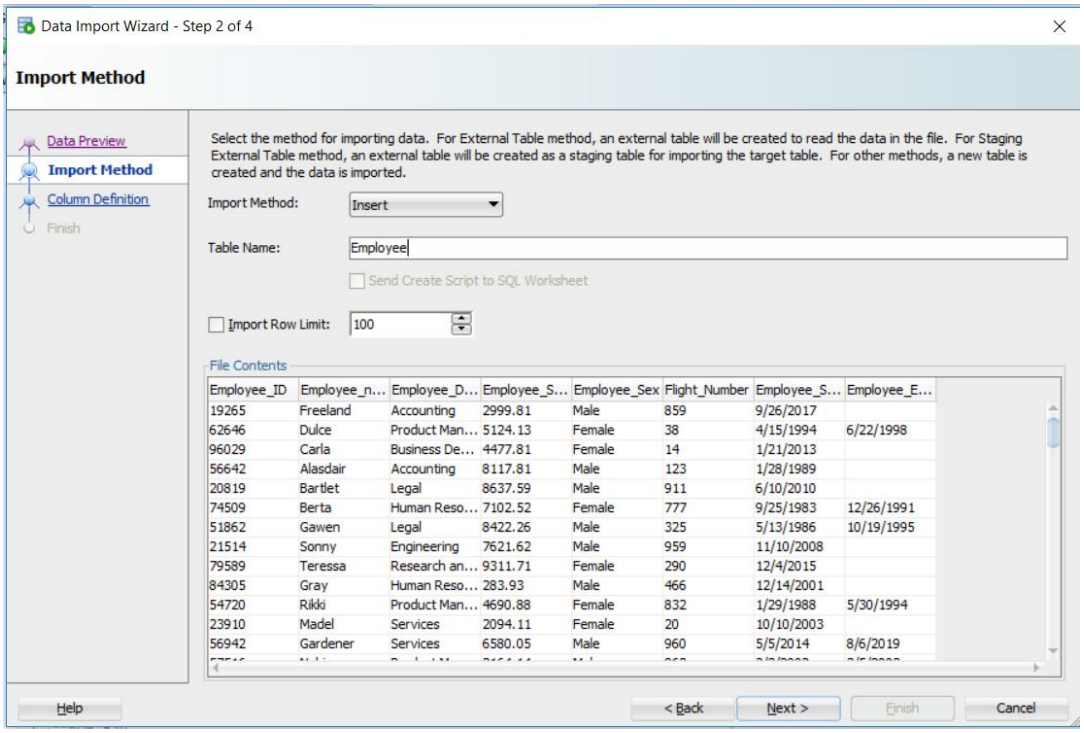


3.4. Import Employee.csv file

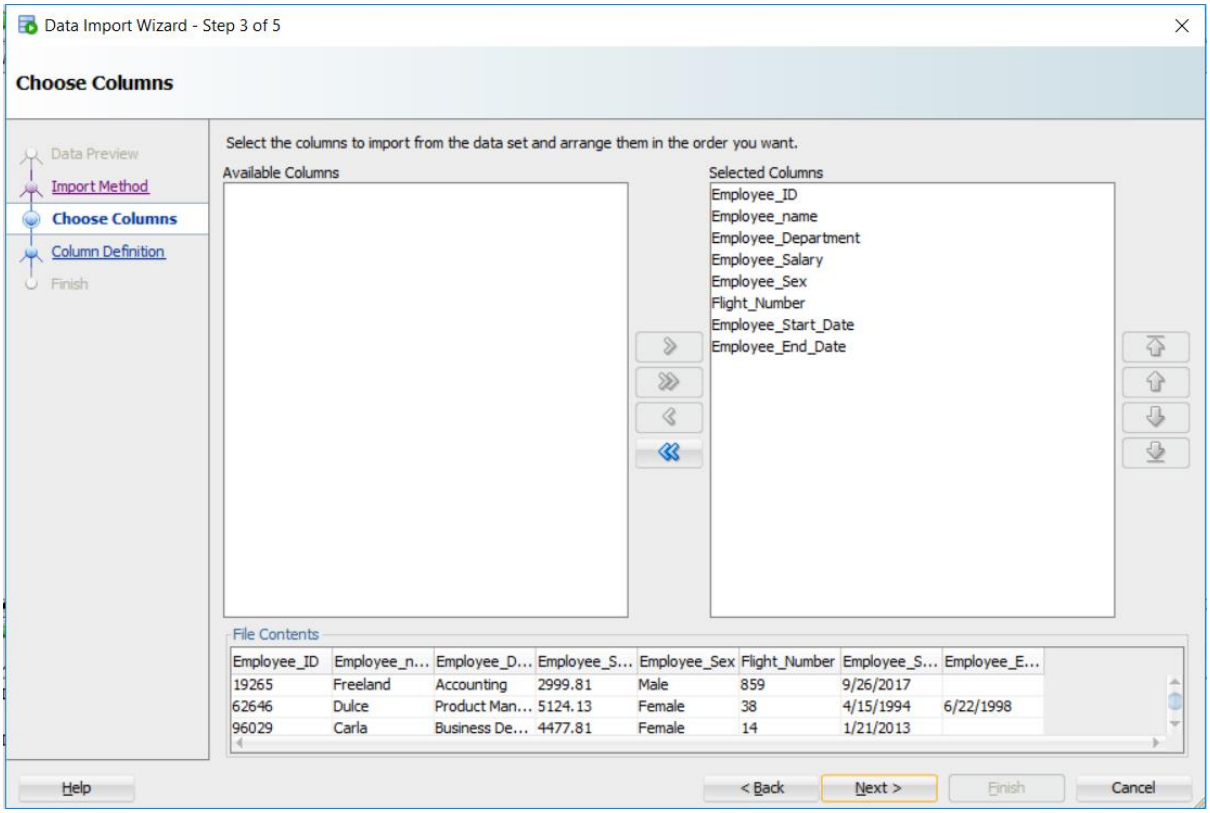
a) Employee.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



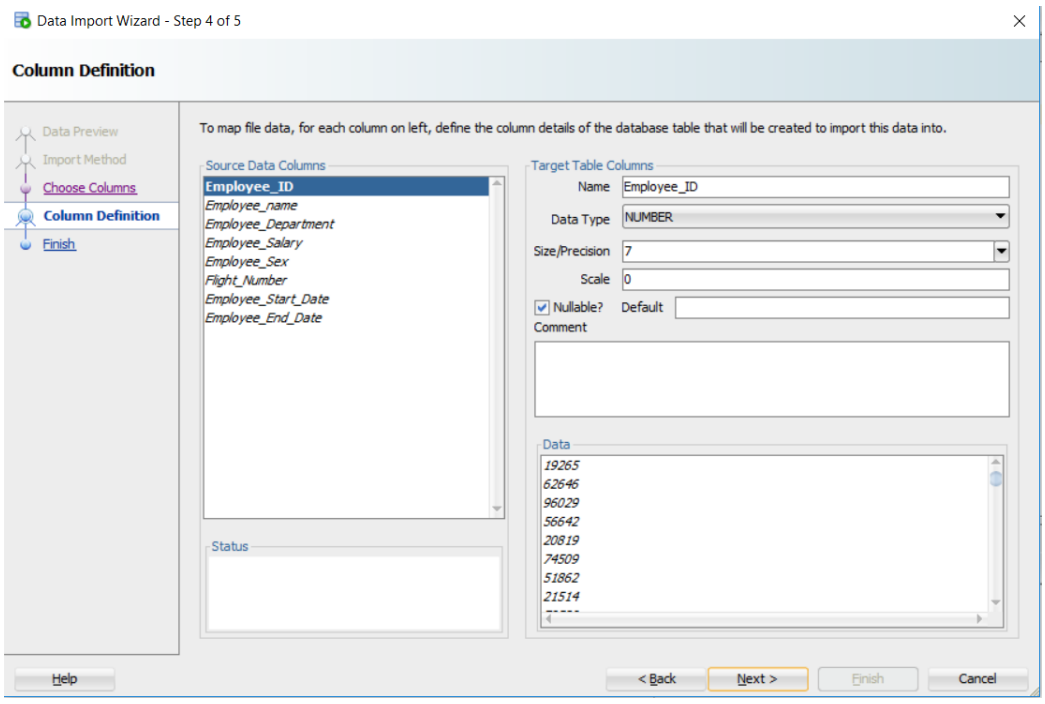
b) Import Method is selected as Insert and provided Table Name as Employee.



c) Columns are selected in Choose Columns tab as below.

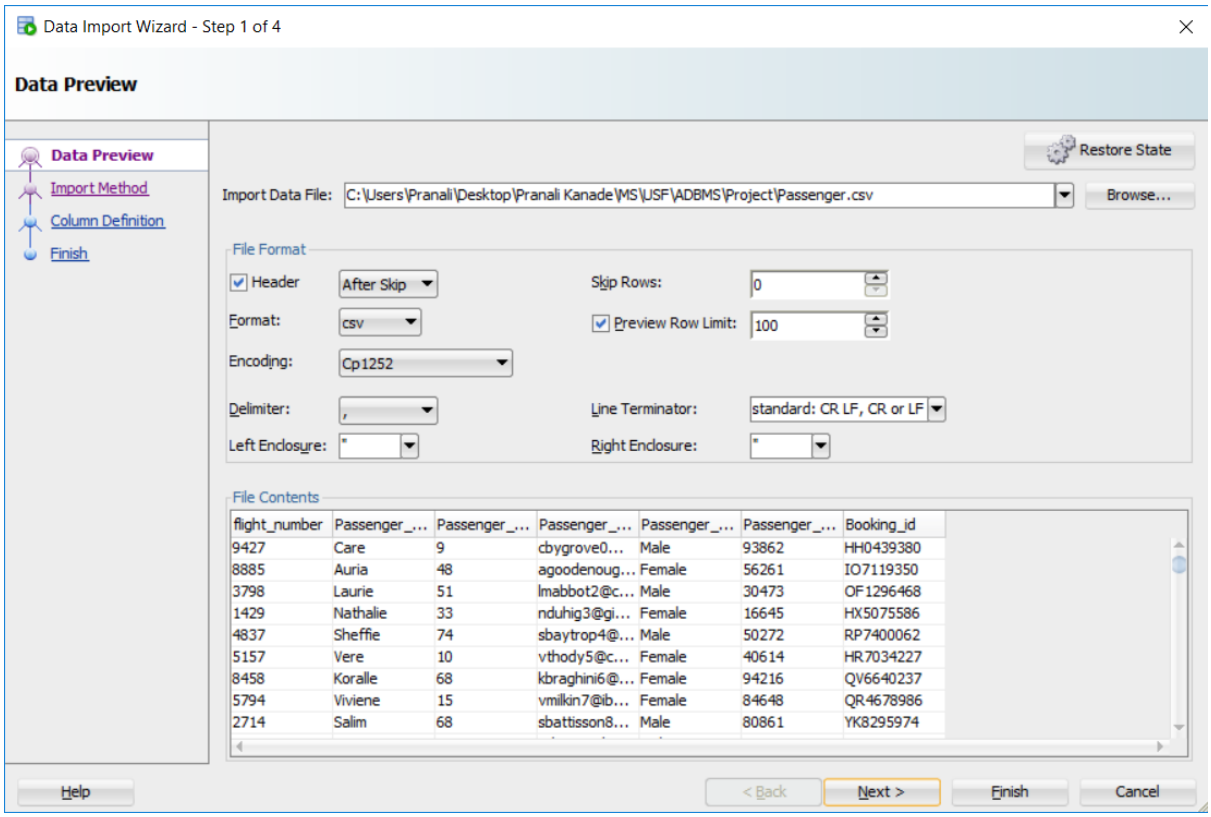


d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

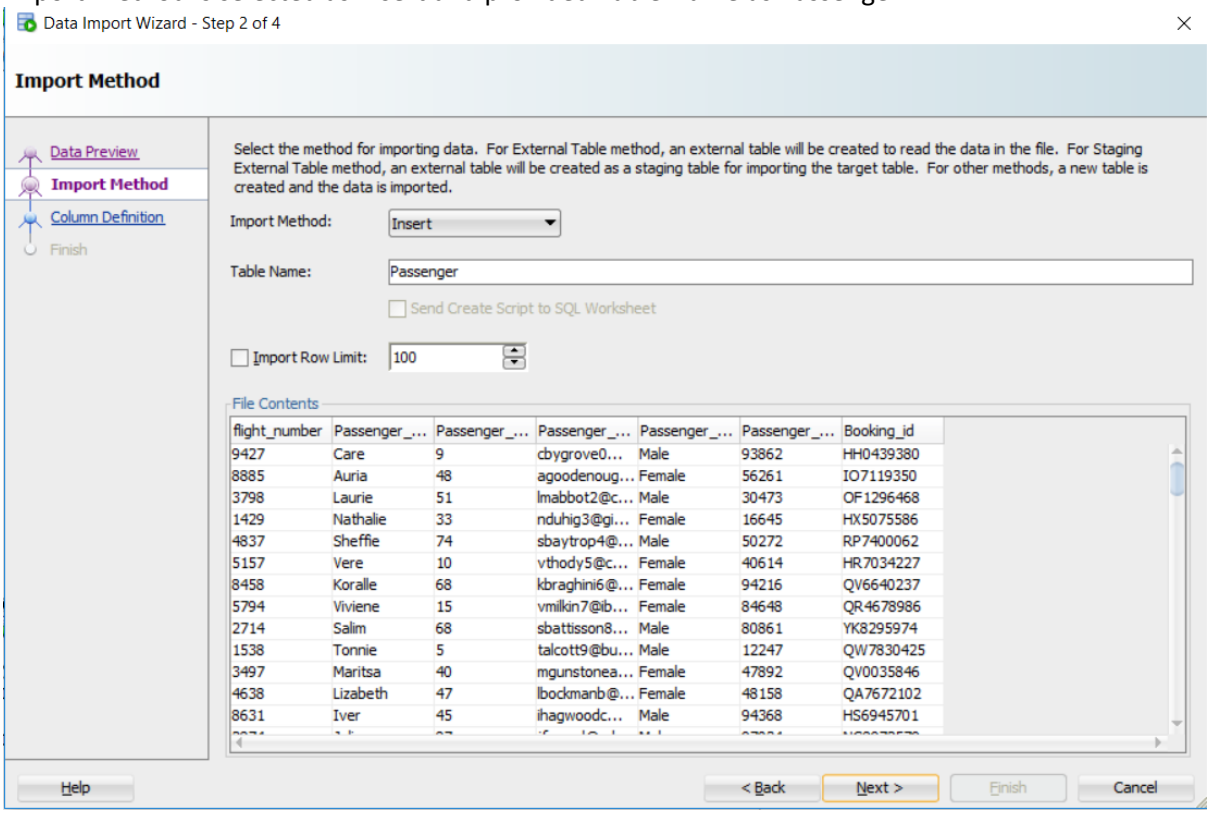


3.5. Import Passenger.csv file

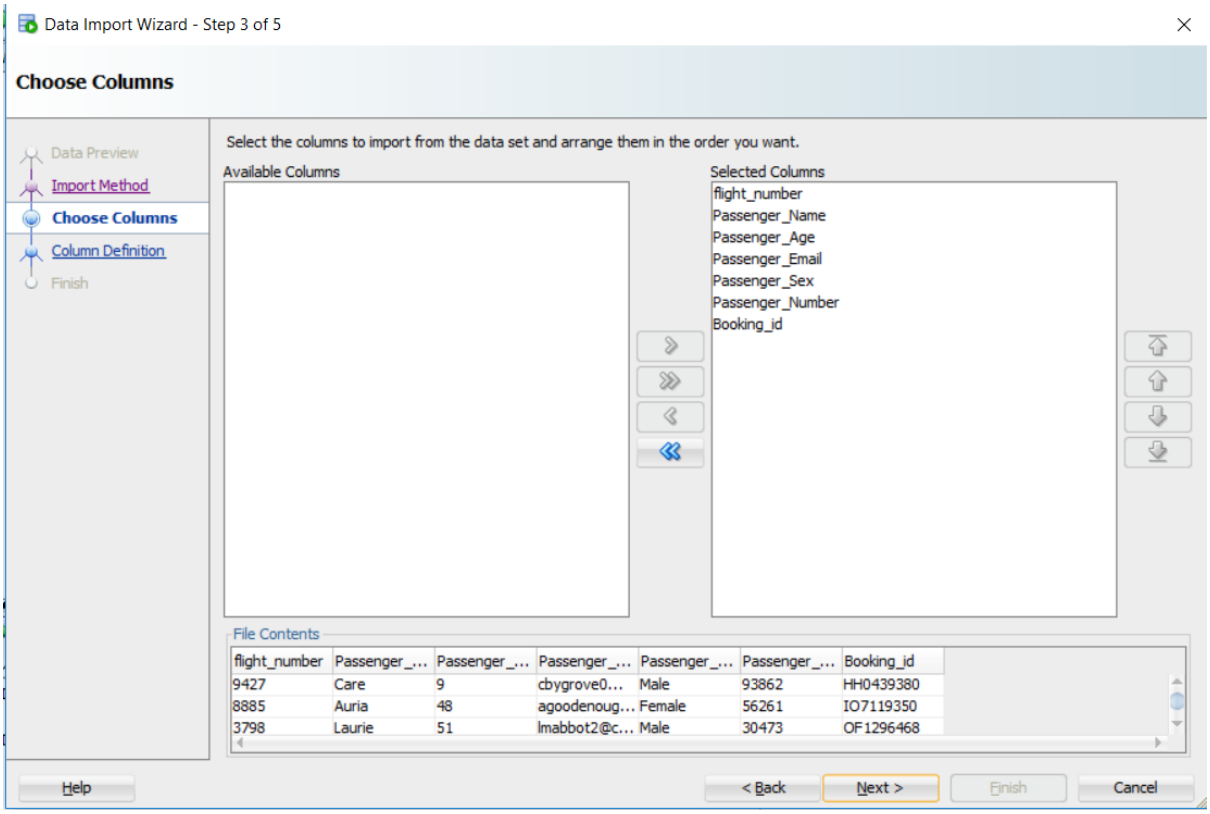
a) Passenger.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



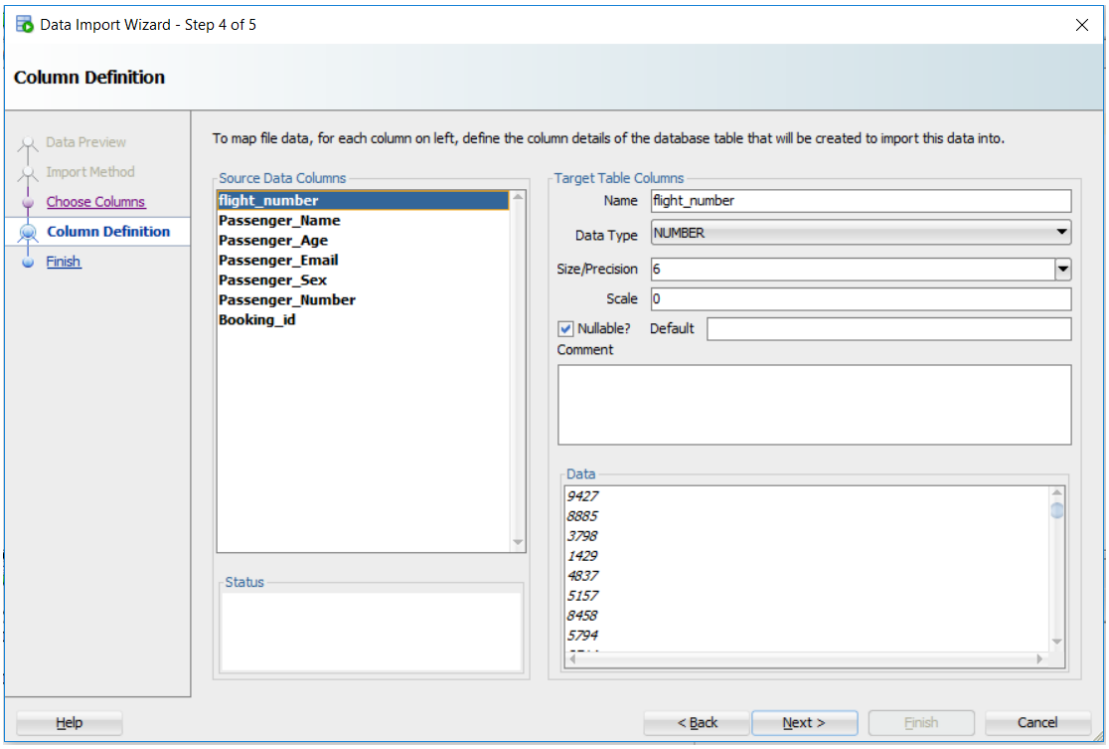
b) Import Method is selected as Insert and provided Table Name as Passenger.



c) Columns are selected in Choose Columns tab as below.

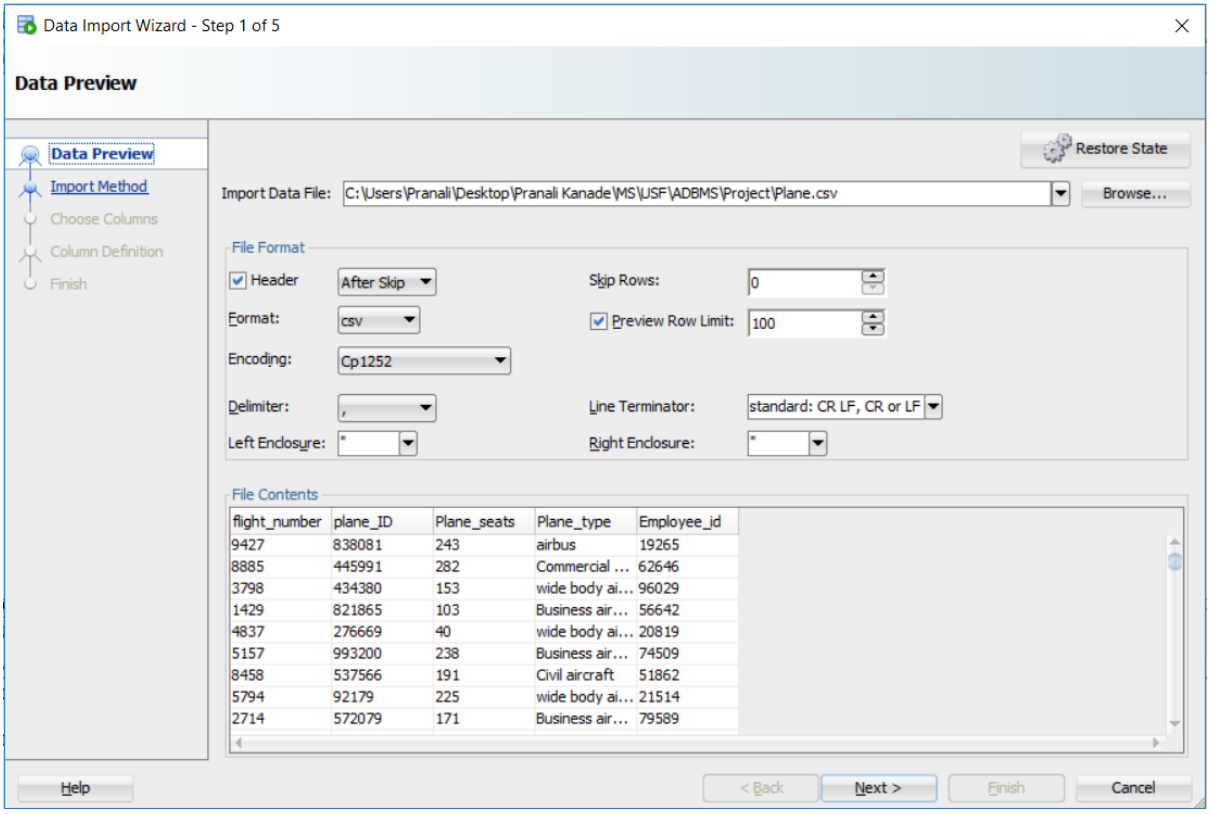


d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

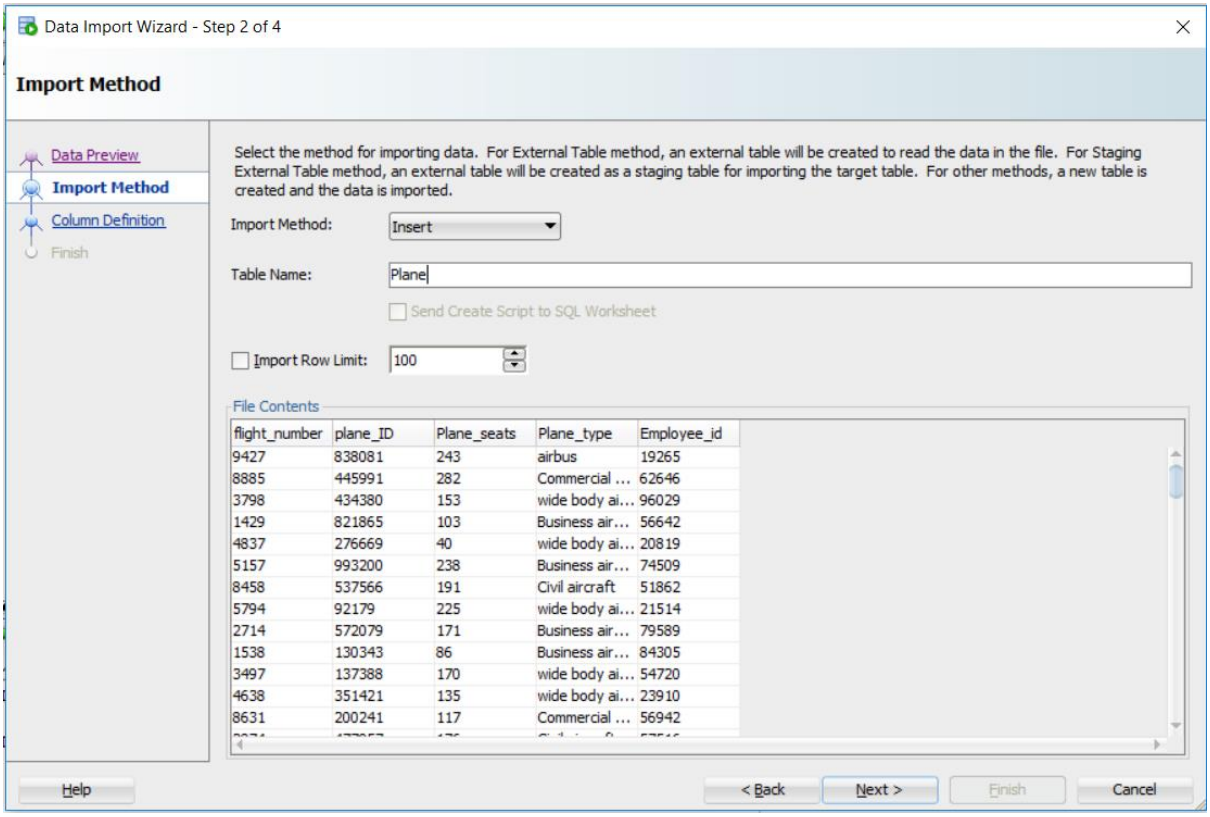


3.6. Import Plane.csv file

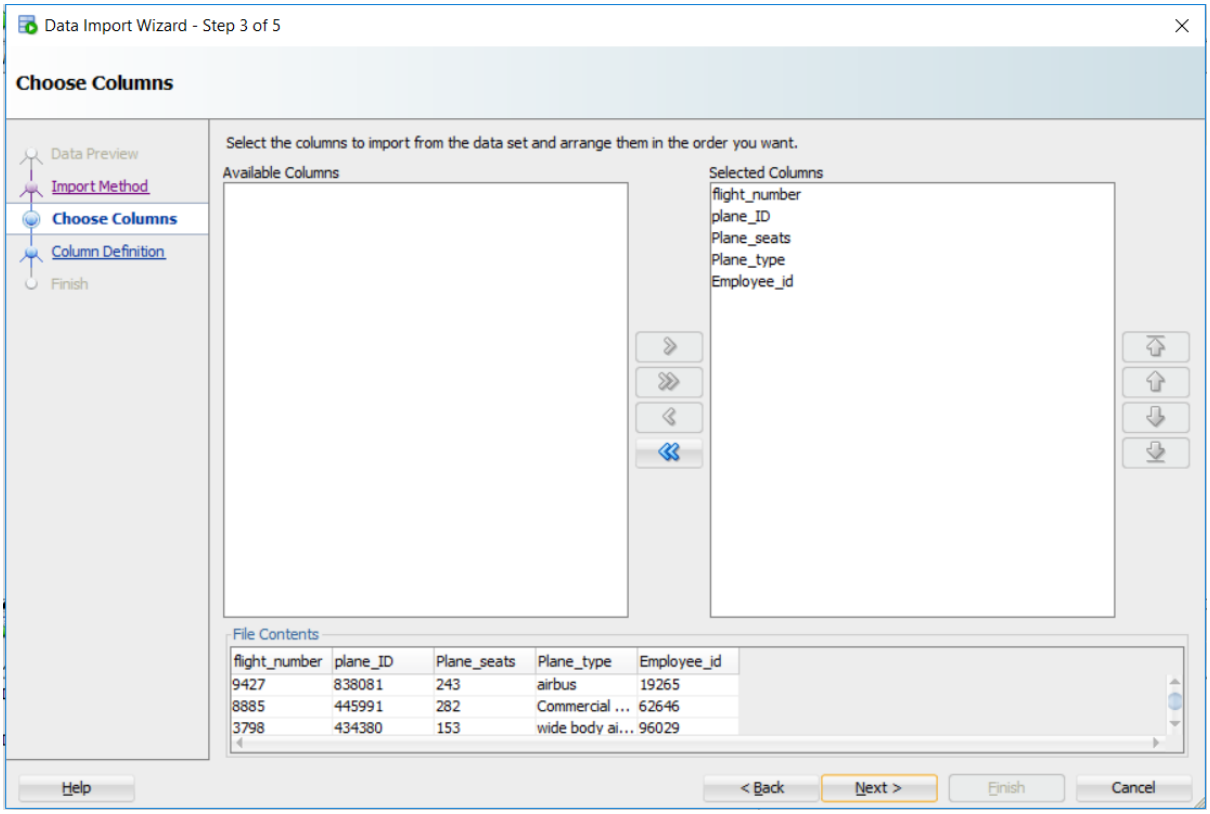
a) Plane.csv is browsed after selecting Data Import option in Oracle SQL Developer as below.



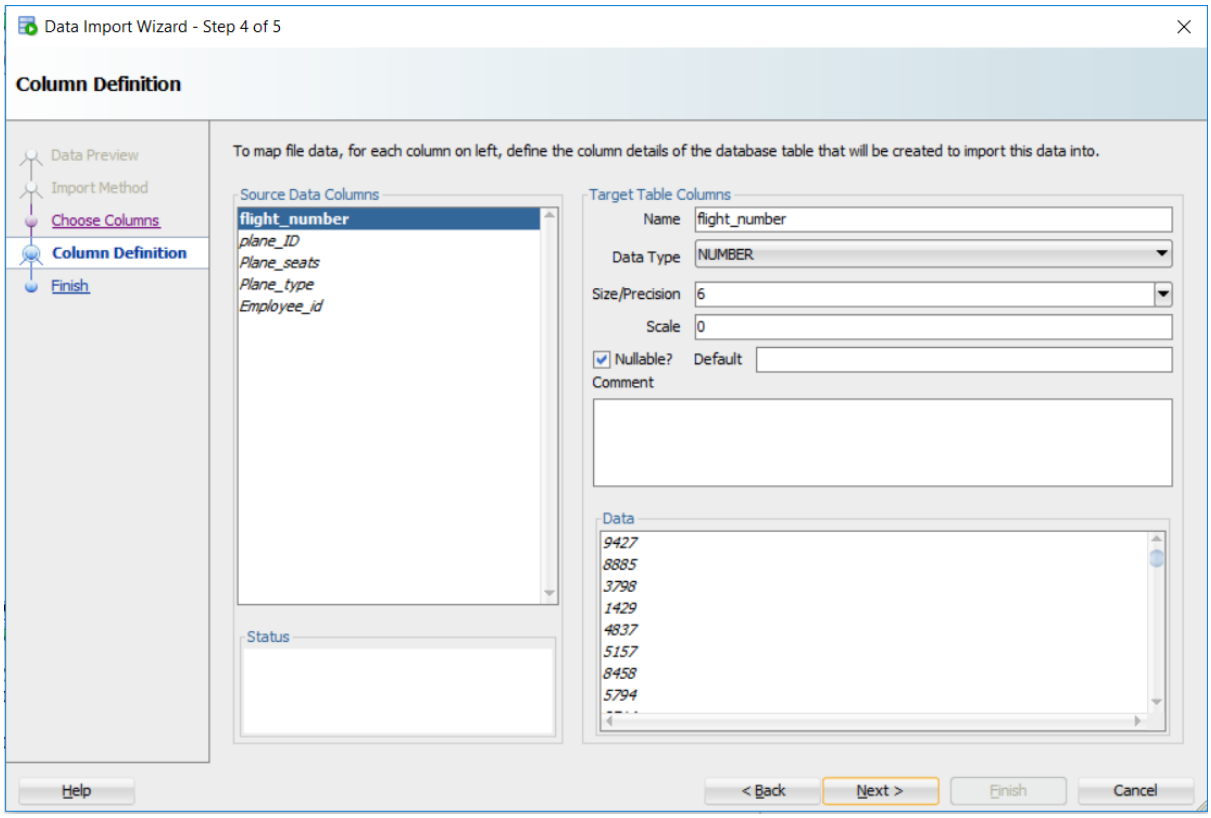
b) Import Method is selected as Insert and provided Table Name as Plane.



c) Columns are selected in Choose Columns tab as below.



d) In Column definition Data Type, Size, Scale and Nullable setting is changed as per columns definition.

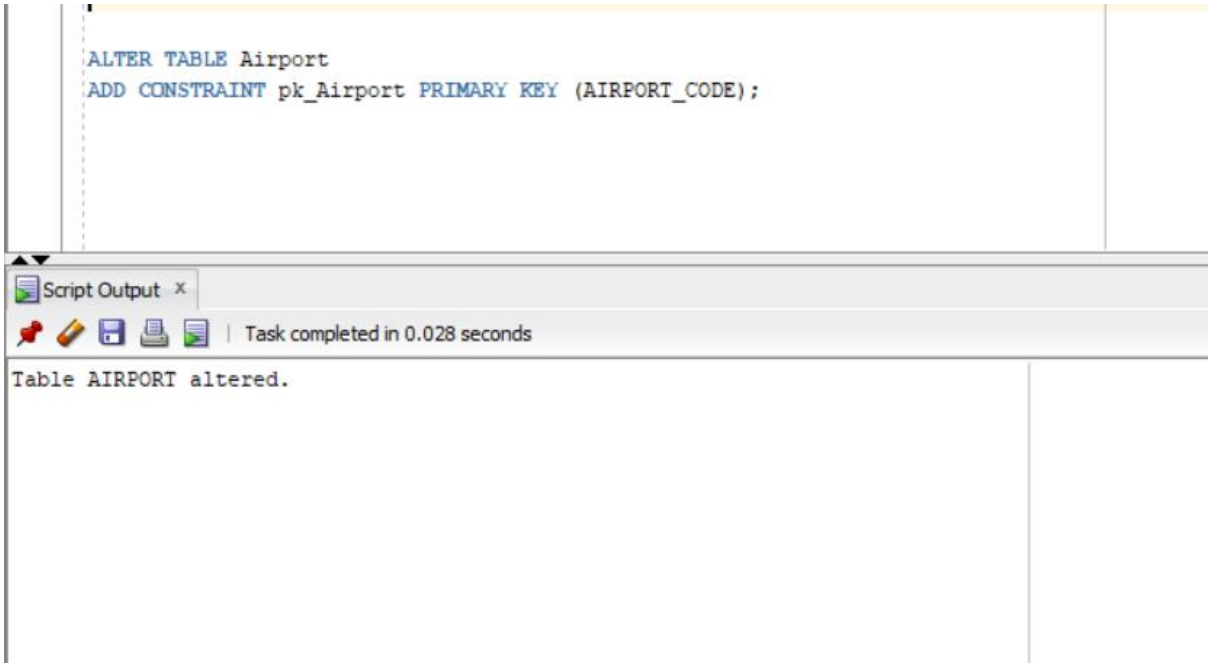


4. Defining primary key and foreign key

4.1. Defining primary key on AIRPORT_CODE column for table Airport

ALTER TABLE Airport

ADD CONSTRAINT pk_Airport PRIMARY KEY (AIRPORT_CODE);

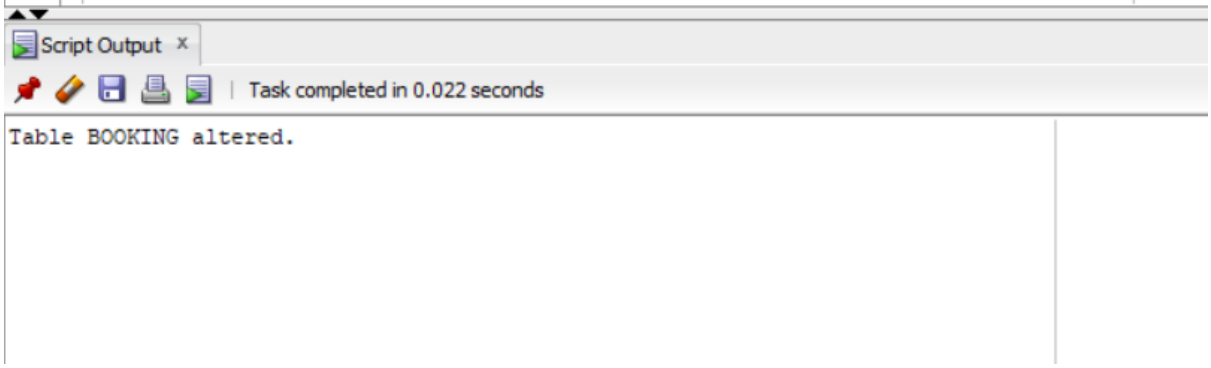


4.2. Defining primary key on Booking_id column for table Booking

ALTER TABLE Booking

ADD CONSTRAINT pk_Booking PRIMARY KEY (Booking_id);

```
ALTER TABLE Booking
ADD CONSTRAINT pk_Booking PRIMARY KEY (Booking_id);
```



4.3. Defining primary key on Flight_Number column for table Flight

ALTER TABLE Flight

ADD CONSTRAINT pk_Flight PRIMARY KEY (Flight_Number);

```
ALTER TABLE Flight
ADD CONSTRAINT pk_Flight PRIMARY KEY (Flight_Number);
```

4.4. Defining primary key on plane_ID column for table Plane

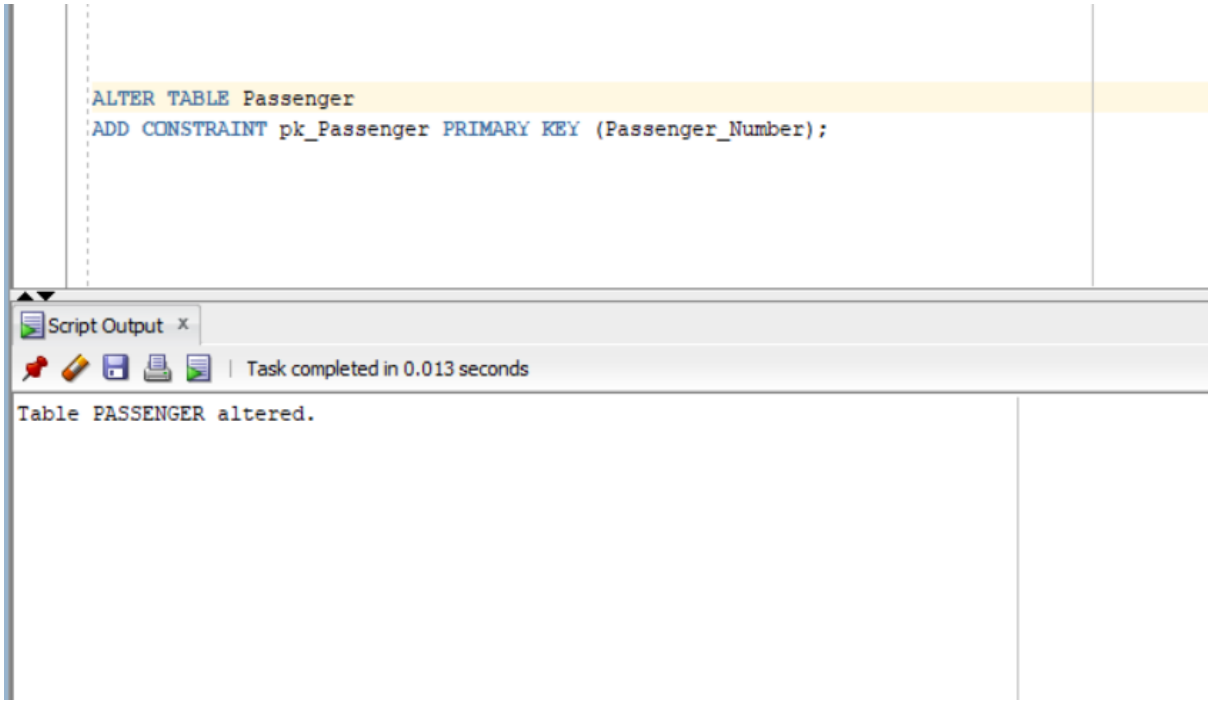
ALTER TABLE Plane

ADD CONSTRAINT pk_Plane PRIMARY KEY (plane_ID);

4.5. Defining primary key on Passenger_Number column for table Passenger

ALTER TABLE Passenger

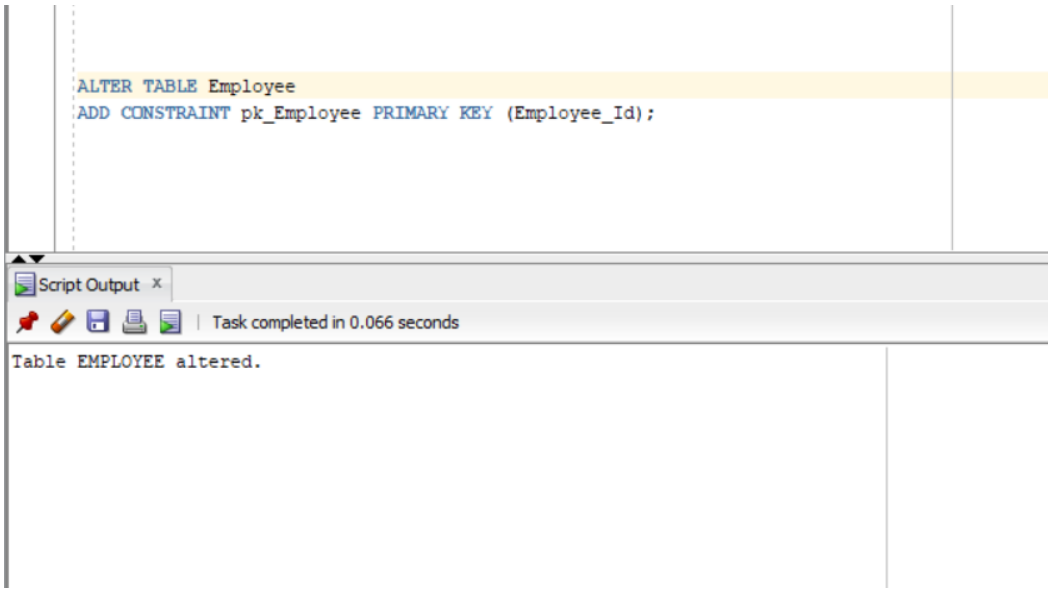
ADD CONSTRAINT pk_Passenger PRIMARY KEY (Passenger_Number);



4.6. Defining primary key on Employee_Id column for table Employee

ALTER TABLE Employee

ADD CONSTRAINT pk_Employee PRIMARY KEY (Employee_Id);



4.7. Foreign key generation on table Flight

ALTER TABLE Flight

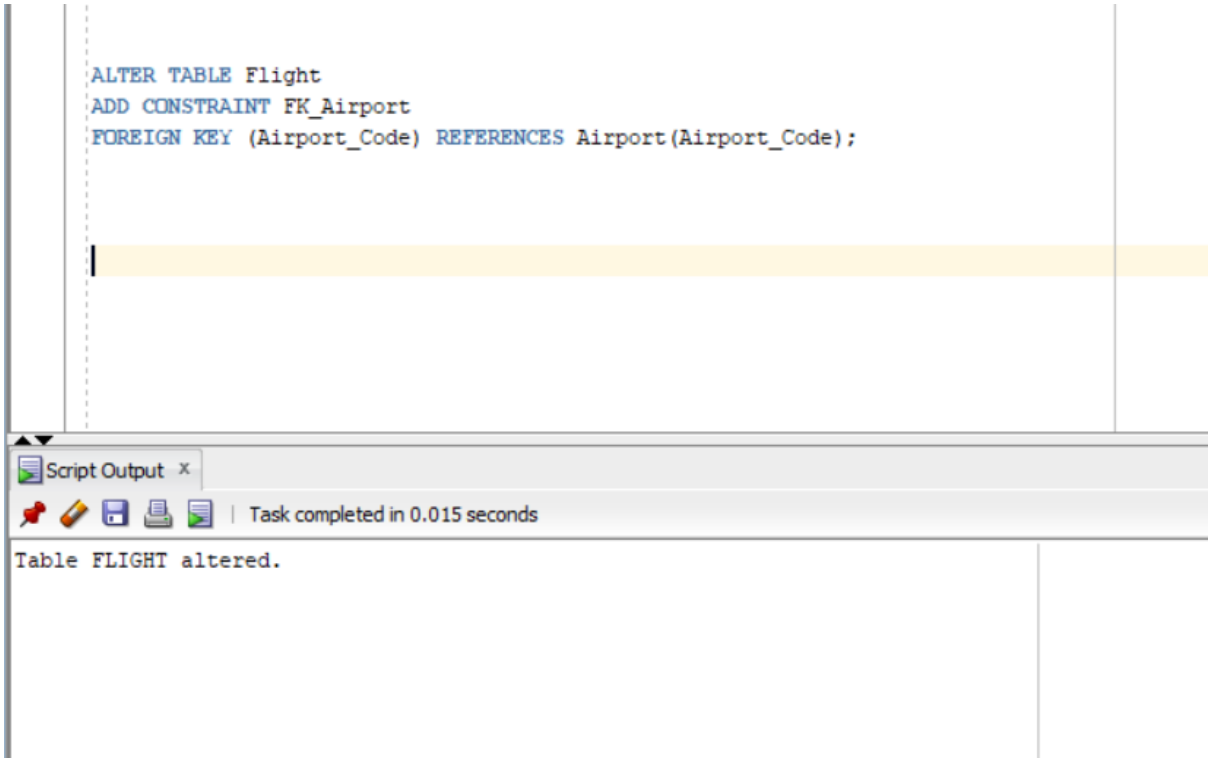
ADD CONSTRAINT FK_Passenger

FOREIGN KEY (Passenger_Number) REFERENCES Passenger(Passenger_Number);

ALTER TABLE Flight

ADD CONSTRAINT FK_Airport

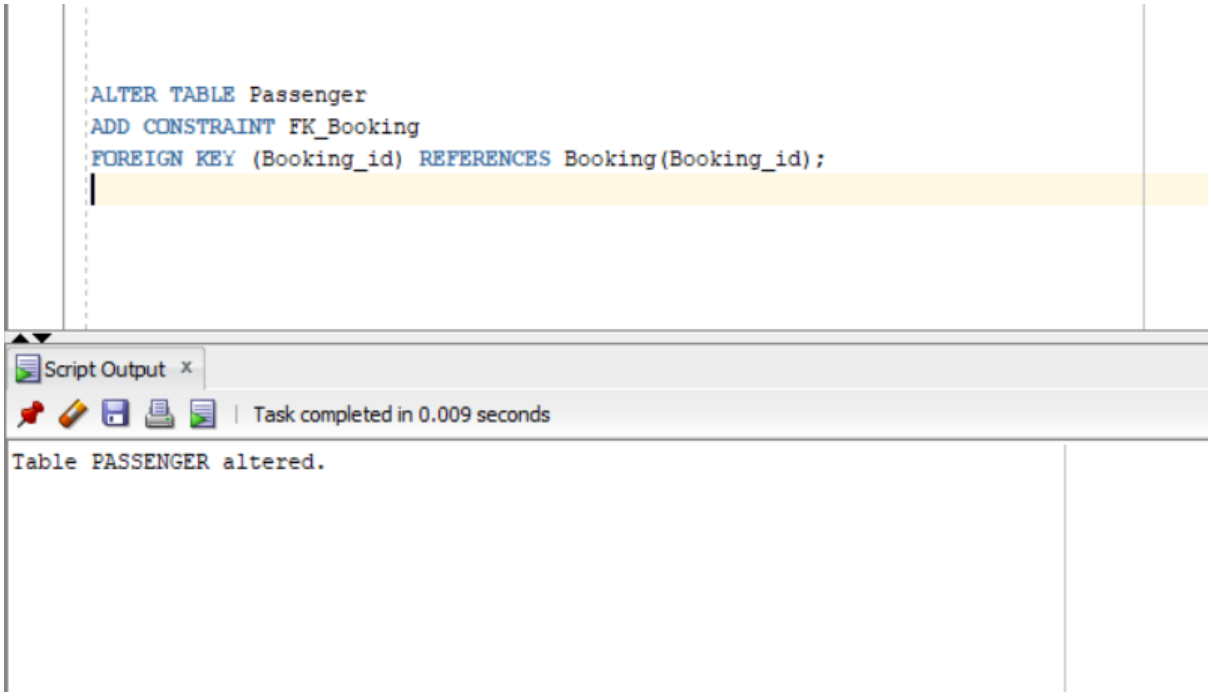
FOREIGN KEY (Airport_Code) REFERENCES Airport(Airport_Code);



ALTER TABLE Passenger

ADD CONSTRAINT FK_Booking

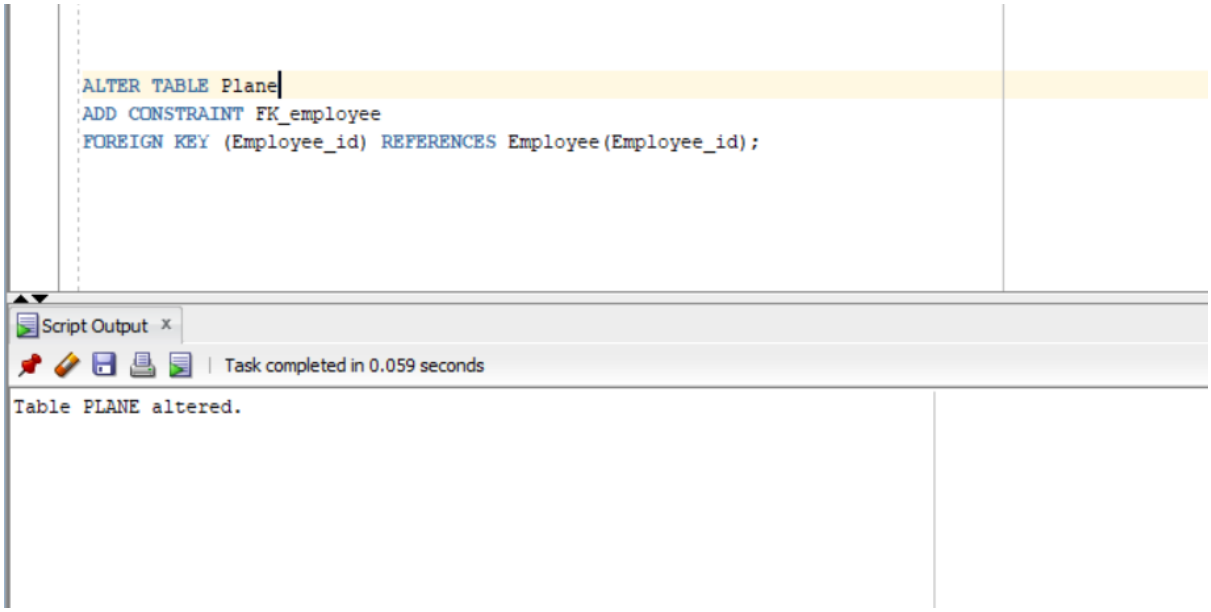
FOREIGN KEY (Booking_id) REFERENCES Booking(Booking_id);



ALTER TABLE Plane

ADD CONSTRAINT FK_employee

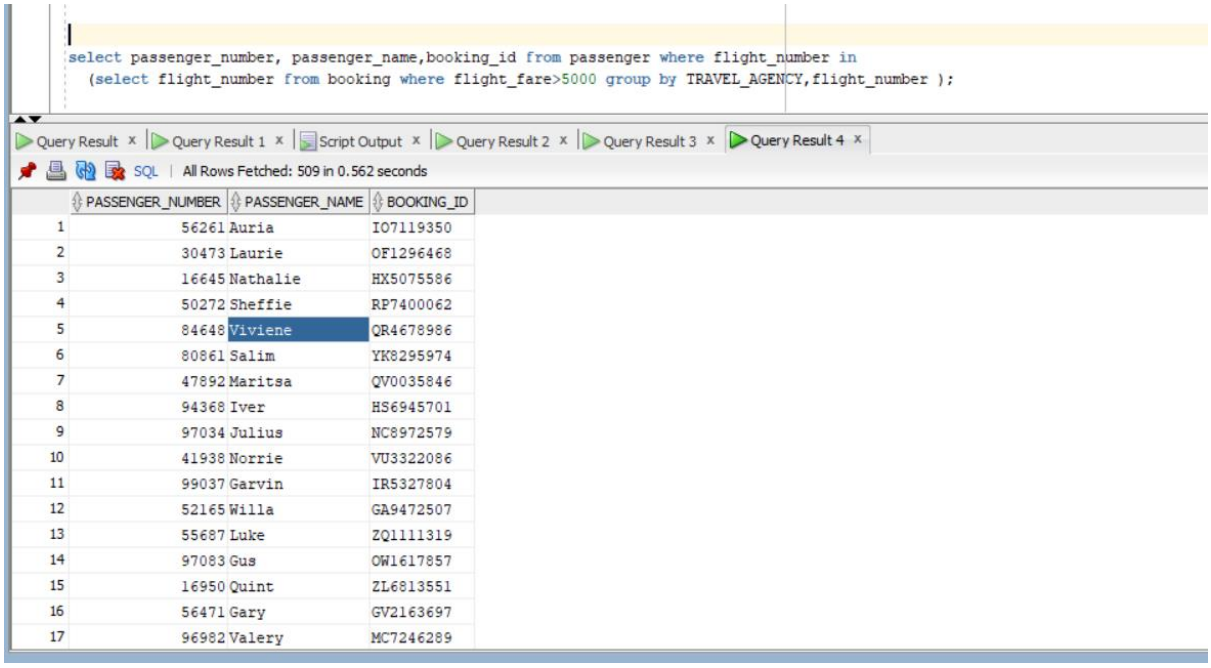
FOREIGN KEY (Employee_id) REFERENCES Employee(Employee_id);



5. Generating queries

1) Generating list of passenger whose flight fare is higher than 5000 depending on travel agency.

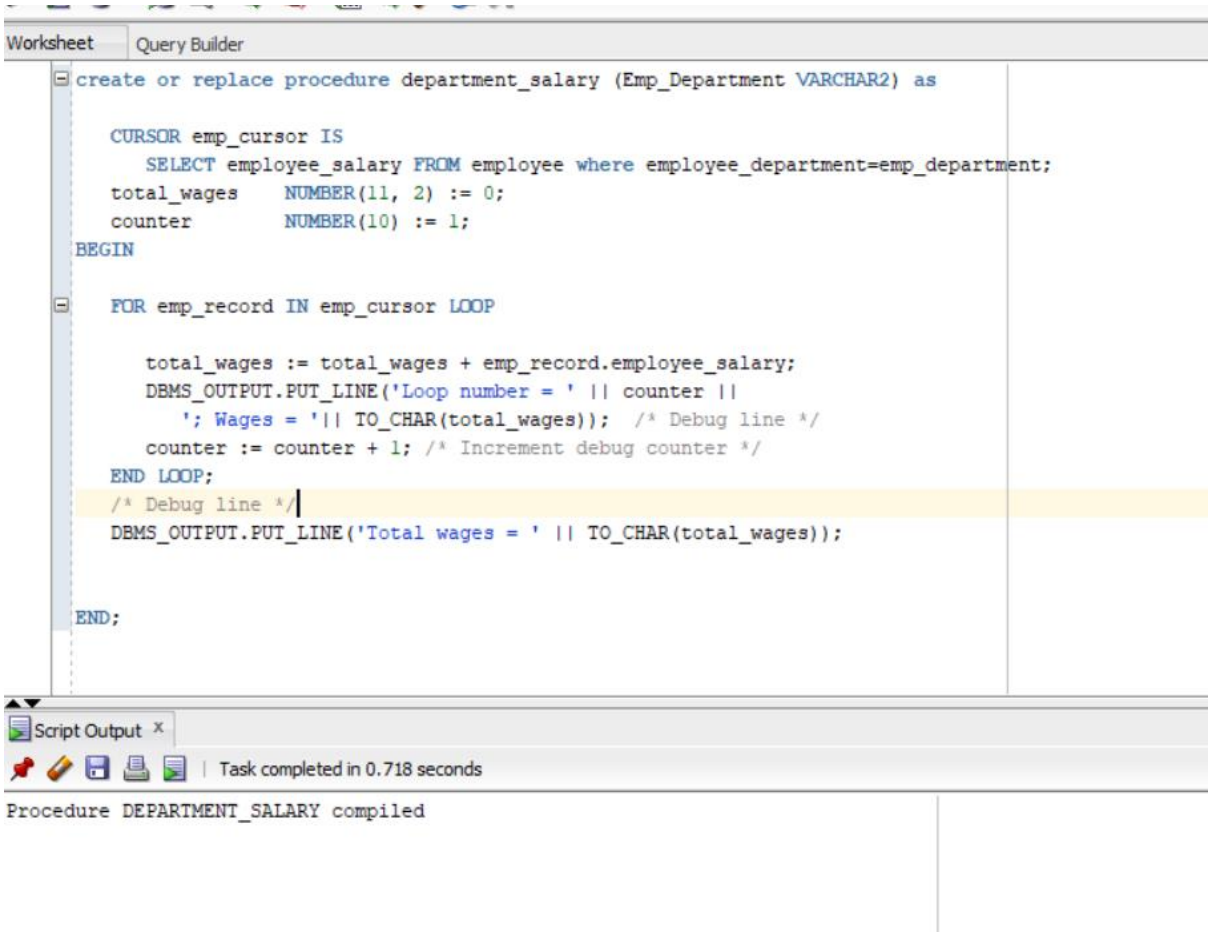
```
select passenger_number, passenger_name,booking_id from passenger where flight_number in
(select flight_number from booking where flight_fare>5000 group by
TRAVEL_AGENCY,flight_number );
```



	PASSENGER_NUMBER	PASSENGER_NAME	BOOKING_ID
1	56261	Auria	IO7119350
2	30473	Laurie	OF1296468
3	16645	Nathalie	HX5075586
4	50272	Sheffie	RP7400062
5	84648	Viviene	QR4678986
6	80861	Salim	YK8295974
7	47892	Maritsa	QV0035846
8	94368	Iver	HS6945701
9	97034	Julius	NC8972579
10	41938	Norrie	VU3322086
11	99037	Garvin	IR5327804
12	52165	Willa	GA9472507
13	55687	Luke	ZQ1111319
14	97083	Gus	OW1617857
15	16950	Quint	ZL6813551
16	56471	Gary	GV2163697
17	96982	Valery	MC7246289

2) Calculating total salary for particular department by creating procedure department_salary by passing employee_department as argument-

```
create or replace procedure department_salary (Emp_Department VARCHAR2) as
CURSOR emp_cursor IS
SELECT employee_salary FROM employee where employee_department=emp_department;
total_wages    NUMBER(11, 2) := 0;
counter        NUMBER(10) := 1;
BEGIN
FOR emp_record IN emp_cursor LOOP
total_wages := total_wages + emp_record.employee_salary;
DBMS_OUTPUT.PUT_LINE('Loop number = ' || counter || '; Wages = '|| TO_CHAR(total_wages));
/* Debug line */
counter := counter + 1; /* Increment debug counter */
END LOOP;
/* Debug line */
DBMS_OUTPUT.PUT_LINE('Total wages = ' || TO_CHAR(total_wages));
END;
```



Execution of procedure department_salary

```
execute department_salary('Accounting');
```

Query Result | Query Result 1 | Script Output | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.299 seconds

```
Loop number = 81; Wages = 442569.32
Loop number = 82; Wages = 448542.43
Loop number = 83; Wages = 448894.65
Loop number = 84; Wages = 458401.6
Loop number = 85; Wages = 463195.13
Loop number = 86; Wages = 468712.53
Loop number = 87; Wages = 476381.31
Loop number = 88; Wages = 477086.69
Loop number = 89; Wages = 482806.89
Loop number = 90; Wages = 486723.21
Loop number = 91; Wages = 487643.15
Loop number = 92; Wages = 492880.12
Loop number = 93; Wages = 494446
Loop number = 94; Wages = 500371.44
Loop number = 95; Wages = 504398.39
Loop number = 96; Wages = 513298.62
Loop number = 97; Wages = 520624.07
Loop number = 98; Wages = 526571.59
Loop number = 99; Wages = 531755.72
Loop number = 100; Wages = 535309.55
Total wages = 535309.55
```

3. Use of inner join query between tables flight and booking to find flight details like average price.

```
select fl.flight_number, fl.flight_source, fl.flight_destination,
round((avg(bk.flight_fare)),2) as average_price
from flight fl  inner join booking bk
on fl.flight_number = bk.flight_number group by fl.flight_number,fl.flight_source,
fl.flight_destination ;
```

```
select fl.flight_number, fl.flight_source, fl.flight_destination,
round((avg(bk.flight_fare)),2) as average_price
from flight fl  inner join booking bk
on fl.flight_number = bk.flight_number group by fl.flight_number,fl.flight_source, fl.flight_destination ;
```

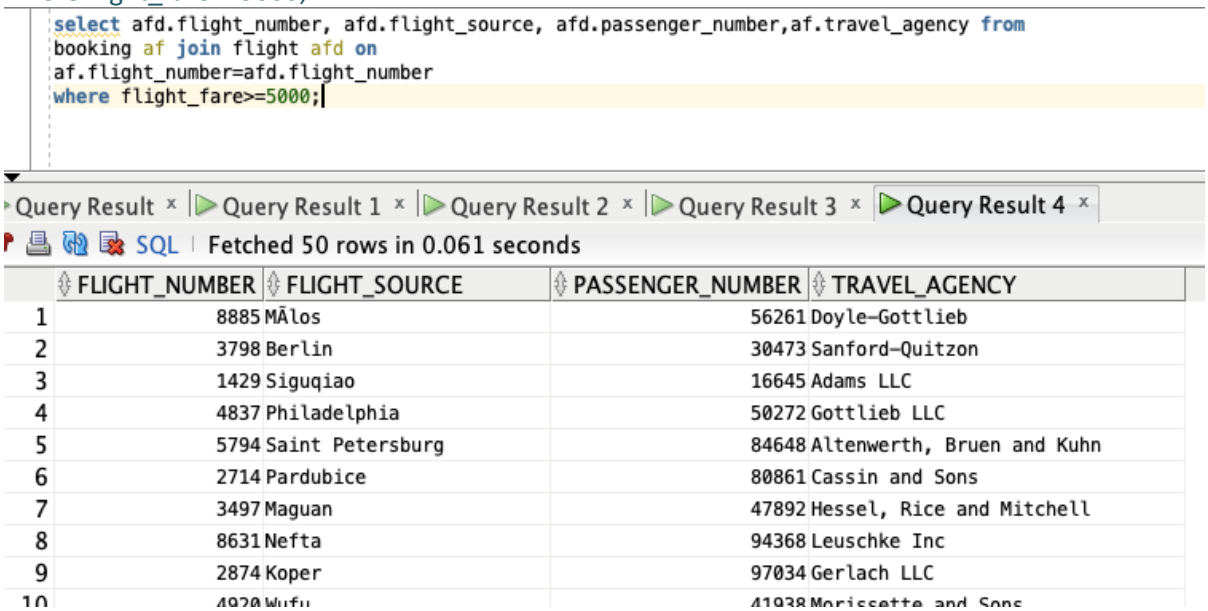
Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | Fetched 50 rows in 0.103 seconds

	FLIGHT_NUMBER	FLIGHT_SOURCE	FLIGHT_DESTINATION	AVERAGE_PRICE
1	4920	Wufu	Osieck	5543.88
2	8363	Bella Vista	Shanggan	318.86
3	7855	Jibu Hulangtu	Novyy Karachay	4698
4	4693	Pedreira	Sanchahe	6301.33
5	5511	AdiakÃ	Ash ShÃ¿mÃyah	7847.32
6	3953	Geoktschai	AsunciÃn Mita	5911
7	3166	UbatÃ	Kijini	3221.07
8	3487	Kuching	Los Angeles	4619.62
9	3782	Hrodna	Qiandian	1804.42

4. Use of join between tables flight and booking to find travel agencies where flight fare is greater than $5000

```
select afd.flight_number, afd.flight_source, afd.passenger_number,af.travel_agency from
booking af join flight afd on
af.flight_number=afd.flight_number
where flight_fare>=5000;
```



Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

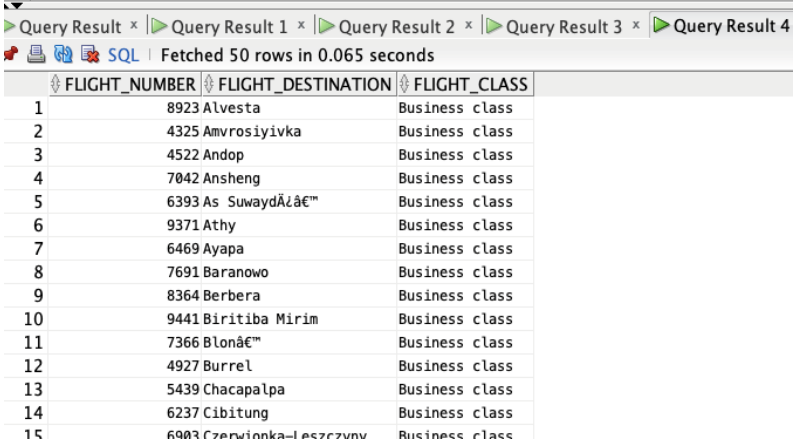
SQL | Fetched 50 rows in 0.061 seconds

	FLIGHT_NUMBER	FLIGHT_SOURCE	PASSENGER_NUMBER	TRAVEL_AGENCY
1	8885	MÃlos	56261	Doyle-Gottlieb
2	3798	Berlin	30473	Sanford-Quitzon
3	1429	Siguqiao	16645	Adams LLC
4	4837	Philadelphia	50272	Gottlieb LLC
5	5794	Saint Petersburg	84648	Altenwerth, Bruen and Kuhn
6	2714	Pardubice	80861	Cassin and Sons
7	3497	Maguan	47892	Hessel, Rice and Mitchell
8	8631	Nefta	94368	Leuschke Inc
9	2874	Koper	97034	Gerlach LLC
10	4920	Wufu	41938	Morissette and Sons

5. Finding the flight class where class is business and flight number is greater than 4000 using most of the functions like where,group by, having, order by.

```
select flight_number, flight_destination,flight_class from flight where flight_class = 'Business class'
group by flight_class,flight_number,flight_destination having flight_number>4000
order by flight_destination;
```

```
select flight_number, flight_destination,flight_class
from flight where flight_class = 'Business class'
group by flight_class,flight_number,flight_destination
having flight_number>4000
order by flight_destination;
```



Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | Fetched 50 rows in 0.065 seconds

	FLIGHT_NUMBER	FLIGHT_DESTINATION	FLIGHT_CLASS
1	8923	Alvesta	Business class
2	4325	Amvrosiyivka	Business class
3	4522	Andop	Business class
4	7042	Ansheng	Business class
5	6393	As SuwaydÄ¿â€™	Business class
6	9371	Athy	Business class
7	6469	Ayapa	Business class
8	7691	Baranowo	Business class
9	8364	Berbera	Business class
10	9441	Biritiba Mirim	Business class
11	7366	Blonâ€™	Business class
12	4927	Burrel	Business class
13	5439	Chacapalpa	Business class
14	6237	Cibitung	Business class
15	6903	Czerwionka-Leszczyny	Business class

6. Use of left join between tables airport and flight to find popularity and source

```
select a.airport_code,a.rank_by_popularity,a.country,f.flight_source
from airport a left join flight f
on a.airport_code=f.airport_code
order by a.rank_by_popularity ;
```

```
select a.airport_code,a.rank_by_popularity,a.country,f.flight_source
from airport a  left join flight f
on a.airport_code=f.airport_code
order by a.rank_by_popularity ;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | Fetched 50 rows in 0.059 seconds

	AIRPORT_CODE	RANK_BY_POPULARITY	COUNTRY	FLIGHT_SOURCE
1	4345	1	Ukraine	Quimper
2	1036	1	Liberia	Ivanovka
3	2736	1	Indonesia	Ruma
4	4877	1	Brazil	Quilino
5	2480	1	Argentina	Hezheng Chengguanzhen
6	8841	1	United States	Tagbacan Ibaba
7	8630	1	Indonesia	Chiang Mai
8	9314	1	Oman	Karanggeneng
9	1889	1	Peru	Zouila
10	3072	1	Philippines	Frederiksberg

7. Use of Left join between tables employee and plane to find name of all female employees who have been on plane of type Boeing.

```
select e.employee_id,e.employee_name from employee e  left join plane p on
e.employee_id=p.employee_id
where e.employee_sex='Female' and p.plane_type = 'Boeing';
```

```
select e.employee_id,e.employee_name from employee e  left join plane p on e.employee_id=p.employee_id
where e.employee_sex='Female' and p.plane_type = 'Boeing';
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | Fetched 50 rows in 0.083 seconds

	EMPLOYEE_ID	EMPLOYEE_NAME
1	62646	Dulce
2	62646	Dulce
3	62646	Dulce
4	62646	Dulce
5	74509	Berta
6	54720	Rikki
7	23910	Madel
8	81703	Shelby
9	10961	Fleur
10	83931	Kanya
11	83931	Kanya
12	83931	Kanya
13	31261	Colette
14	22843	Adriaens
15	22843	Adriaens

8. Use or Left Join between tables flight and airport to see flight destination, source based on the rank by popularity.

select f.flight_source, f.flight_destination ,a.rank_by_popularity
from flight f left join airport a
on a.airport_code=f.airport_code
order by rank_by_popularity;

```
select f.flight_source, f.flight_destination ,a.rank_by_popularity
from flight f left join  airport a
on a.airport_code=f.airport_code
order by rank_by_popularity;
```

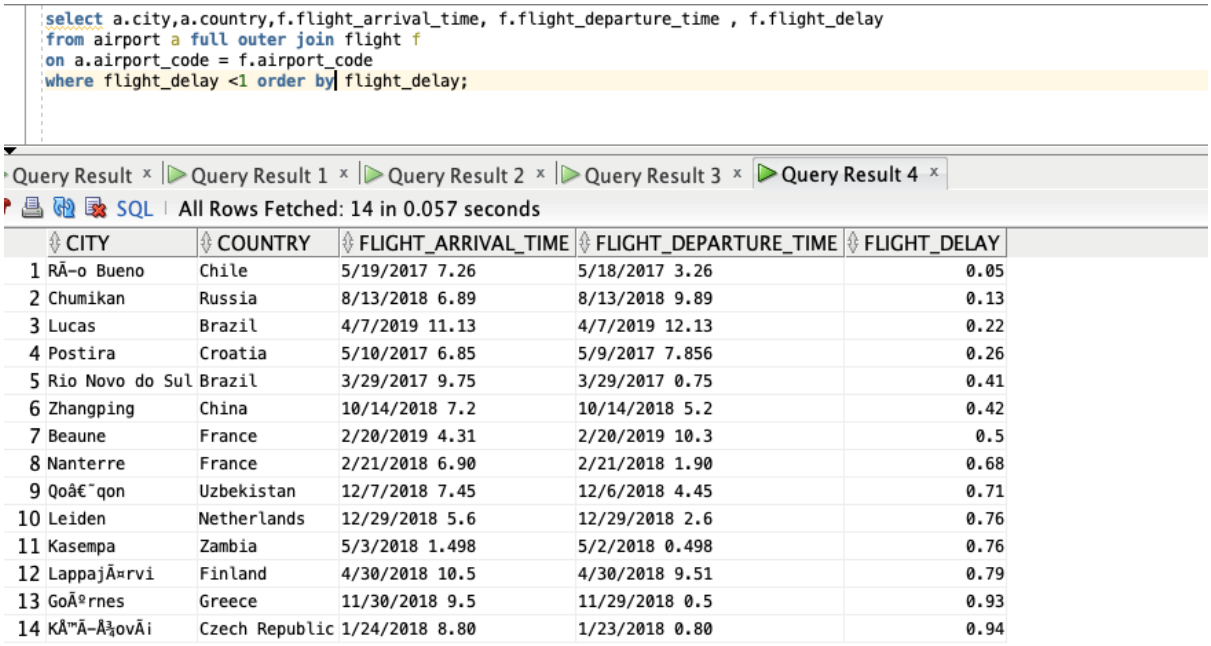
Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result

SQL | Fetched 50 rows in 0.06 seconds

	FLIGHT_SOURCE	FLIGHT_DESTINATION	RANK_BY_POPULARITY
1	Quimper	Shazhenxi	1
2	Ivanovka	Lope de Vega	1
3	Frederiksberg	Lyubar	1
4	La Roxas	Xinxikou	1
5	Flagstaff	Meringkik	1
6	Qianzhou	Diyarb Najm	1
7	Ruma	Jamalteca	1
8	Quilino	Dunhua	1
9	Hezheng Chengguanzhen	AsprÃingeloi	1
10	Tagbacan Ibaba	Cicayur	1
11	Kubangwaru	Qingquan	1
12	San Pedro	Sukasada	1
13	Santa CecÃ-lia	Monte Novo	1
14	Chiang Mai	Yanjiang	1
15	Karanggeneng	Wensu	1
16	Zouila	Zouila	1
17	Khyzy	Boston	1
18	ZajeÄ¿ar	Orizari	1
19	Tours	ColÃn	1
20	Koper	Bagratashen	2

9. Use of Full Outer Join between tables flight and airport to see the city,country whose flights have delayed by less than 1 hour.

```
select a.city,a.country,f.flight_arrival_time, f.flight_departure_time , f.flight_delay
from airport a full outer join flight f
on a.airport_code = f.airport_code
where flight_delay <1 order by flight_delay;
```



```
select a.city,a.country,f.flight_arrival_time, f.flight_departure_time , f.flight_delay
from airport a full outer join flight f
on a.airport_code = f.airport_code
where flight_delay <1 order by flight_delay;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 14 in 0.057 seconds

	CITY	COUNTRY	FLIGHT_ARRIVAL_TIME	FLIGHT_DEPARTURE_TIME	FLIGHT_DELAY
1	RÃ–o Bueno	Chile	5/19/2017 7.26	5/18/2017 3.26	0.05
2	Chumikan	Russia	8/13/2018 6.89	8/13/2018 9.89	0.13
3	Lucas	Brazil	4/7/2019 11.13	4/7/2019 12.13	0.22
4	Postira	Croatia	5/10/2017 6.85	5/9/2017 7.856	0.26
5	Rio Novo do Sul	Brazil	3/29/2017 9.75	3/29/2017 0.75	0.41
6	Zhangping	China	10/14/2018 7.2	10/14/2018 5.2	0.42
7	Beaune	France	2/20/2019 4.31	2/20/2019 10.3	0.5
8	Nanterre	France	2/21/2018 6.90	2/21/2018 1.90	0.68
9	Qoâ€˜qon	Uzbekistan	12/7/2018 7.45	12/6/2018 4.45	0.71
10	Leiden	Netherlands	12/29/2018 5.6	12/29/2018 2.6	0.76
11	Kasempa	Zambia	5/3/2018 1.498	5/2/2018 0.498	0.76
12	LappajÃ¤rvi	Finland	4/30/2018 10.5	4/30/2018 9.51	0.79
13	GoÃºrnes	Greece	11/30/2018 9.5	11/29/2018 0.5	0.93
14	KÅ™Ã–Å¾ovÃ¡	Czech Republic	1/24/2018 8.80	1/23/2018 0.80	0.94

10. Use of 3 queries using inner join between tables flight,plane,booking to see plane type, class and fare

```
select plane_type,flight_class,flight_fare
from( plane p inner join flight f on p.flight_number = f.flight_number)
inner join booking b
on f.flight_number=b.flight_number;
```

```
select plane_type,flight_class,flight_fare
from( plane p inner join flight f
on p.flight_number = f.flight_number)
inner join booking b
on f.flight_number=b.flight_number;
```

Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | Fetched 50 rows in 0.086 seconds

	PLANE_TYPE	FLIGHT_CLASS	FLIGHT_FARE
1	airbus	Business Class	2037.29
2	Commercial airliner	Business class	7398.34
3	wide body airliner	Complimentary upgrades	7415.37
4	Business aircraft	Complimentary upgrades	9574.18
5	wide body airliner	Business Class	7662.41
6	Business aircraft	Premium Economy	3914.48
7	Civil aircraft	Premium Economy	1166.21
8	wide body airliner	Complimentary upgrades	8604.81
9	Business aircraft	Premium Economy	5799.33
10	Business aircraft	Discounted Economy	1551.49
11	wide body airliner	Business Class	7306.48
12	wide body airliner	Discounted Economy	3078.5
13	Commercial airliner	Business Class	7444.05

6. Optimization

6.1. Performance Tuning

- **Parallel Processing**

Parallel Query **processing** in databases systems means that the job of producing the results of the query gets divided between many processes which execute in **parallel**, thus leading to improvements in performance.

- **Degree of Parallelism**

The number of parallel execution servers associated with a single operation is known as the **degree of parallelism** (DOP). Parallel execution is designed to effectively use multiple CPUs. Oracle Database parallel execution framework enables you to either explicitly choose a specific degree of parallelism or to rely on Oracle Database to automatically control it.

Advantages of Parallel Processing:

- **Speed:**

  The main advantage to parallel databases is speed. The server breaks up a user database request into parts and dispatches each part to a separate computer. They work on the parts simultaneously and merge the results, passing them back to the user. This speeds up most data requests, allowing faster access to very large databases.

- **Reliability:**

  A parallel database, properly configured, can continue to work despite the failure of any computer in the cluster.

- **Capacity:**

  As more users request access to the database, the computer administrators add more computers to the parallel server, boosting its overall capacity.

7. Purpose of the experiment:

In this experiment we tried to improve the performance of our database system by bringing in the concepts of Parallel Query Processing. The whole idea was to check if the cost of executing of a query gets affected by parallel processing.

7.1. Steps followed to run the experiment:

1. **First we created new tables so as to perform our execution on those new tables.**

```
SELECT * FROM BOOKING_DATA;

CREATE TABLE BOOKING_DATA AS

SELECT * FROM BOOKING_DATA;

SELECT * FROM BOOKING_DATA;

CREATE TABLE FLIGHT_DATA AS

SELECT * FROM FLIGHT;

SELECT * FROM FLIGHT_DATA;
```

2. **Secondly, we tried executing a query and looked at results without Parallelism. Here the cost of execution was noted by looking at the explain plan, which was way high.**

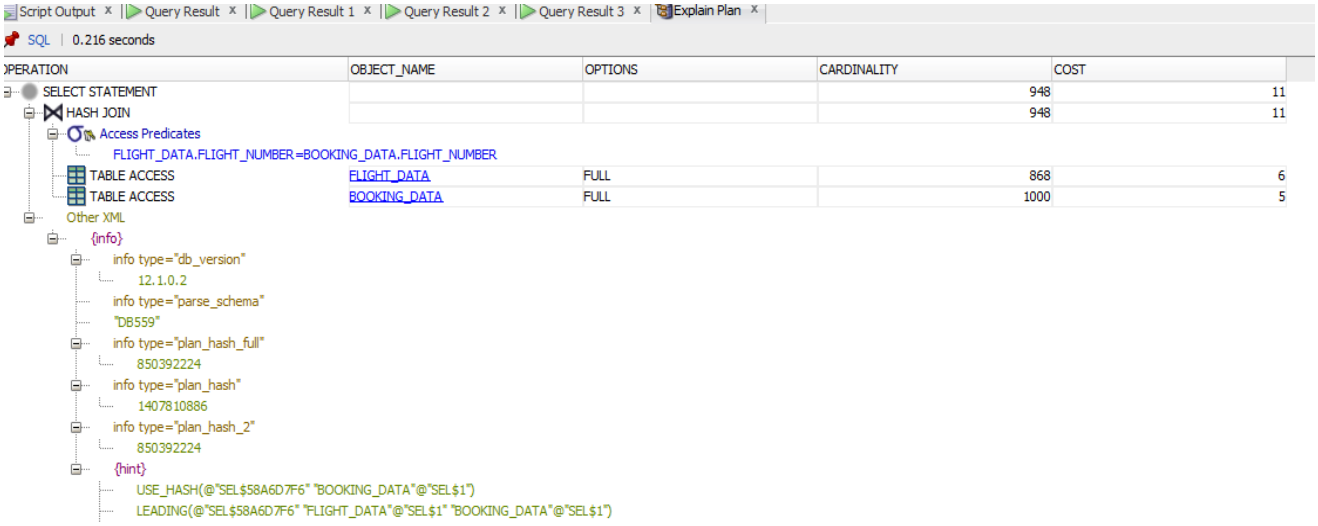
```
ALTER TABLE booking_data NOPARALLEL;

ALTER TABLE flight_data NOPARALLEL;
```

3. **Thirdly we ran a query which fetches records of a passengers booking id and class of the seat that he/she booked.**

```
SELECT flight_class,booking_id

FROM flight_data

INNER JOIN booking_data
```

```
ON flight_data.flight_number = booking_data.flight_number;
```

EXPLAIN PLAN FOR THE ABOVE:



As seen in the above explain plan the cost is 11 and 6 which we will look to reduce by Parallel Processing and hence improve the functioning of our database system.

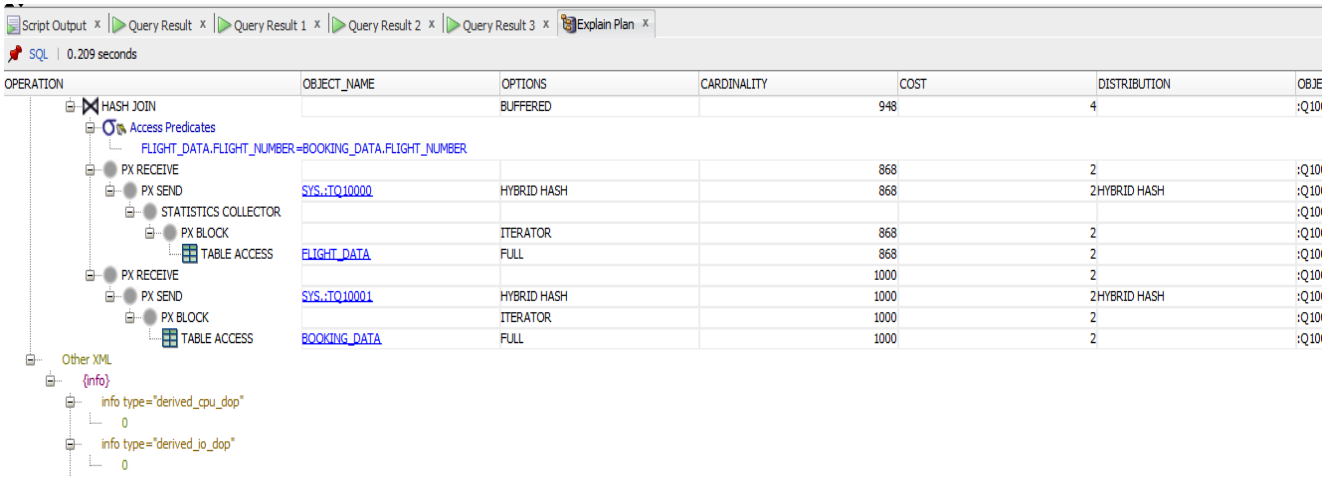
4. **Now we introduced parallel processing and assigned degree of parallelism to the tables. Which means how many parallel executing servers are associated with a single operation. Here we kept the DOP to be**

```
ALTER TABLE booking_data PARALLEL 3;

ALTER TABLE flight_data PARALLEL 3;
```

5. **Now we executed the same query but with assigned parallelism and then looked at explain plan.**

```
SELECT flight_class,booking_id

FROM flight_data

INNER JOIN booking_data

ON flight_data.flight_number = booking_data.flight_number;
```

EXPLAIN PLAN FOR ABOVE:



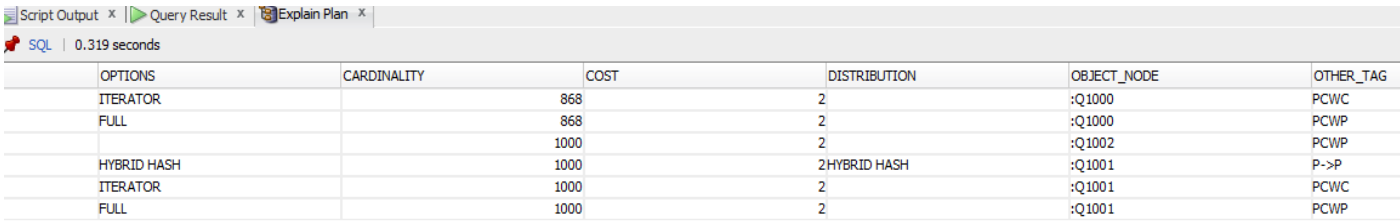
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	DISTRIBUTION	OBJE
HASH JOIN		BUFFERED	948	4		:Q10
Access Predicates						
FLIGHT_DATA.FLIGHT_NUMBER=BOOKING_DATA.FLIGHT_NUMBER						
PX RECEIVE			868	2		:Q10
PX SEND	SYS.:TQ10000	HYBRID HASH	868	2	HYBRID HASH	:Q10
STATISTICS COLLECTOR						:Q10
PX BLOCK		ITERATOR	868	2		:Q10
TABLE ACCESS	FLIGHT_DATA	FULL	868	2		:Q10
PX RECEIVE			1000	2		:Q10
PX SEND	SYS.:TQ10001	HYBRID HASH	1000	2	HYBRID HASH	:Q10
PX BLOCK		ITERATOR	1000	2		:Q10
TABLE ACCESS	BOOKING_DATA	FULL	1000	2		:Q10
Other XML						
{info}						
info type="derived_cpu_dop"						
0						
info type="derived_io_dop"						
0						

As seen in the above explain plan the cost gets significantly reduced after Parallel Processing is introduced. The cost gets reduced to 4 and 2 and all the operators like PX coordinator and PCWP operator for the parallel execution are getting used.

6. **Now looking at the explain plan for DOP-3 we then further increased the Parallel processing servers to 5 and then checked if the cost is getting reduced even further.**

```
ALTER TABLE booking_data PARALLEL 5;

ALTER TABLE flight_data PARALLEL 5;


SELECT flight_class,booking_id

FROM flight_data

INNER JOIN booking_data

ON flight_data.flight_number = booking_data.flight_number;
```

EXPLAIN PLAN FOR:


Script Output | Query Result | Explain Plan

SQL | 0.319 seconds

	OPTIONS	CARDINALITY	COST	DISTRIBUTION	OBJECT_NODE	OTHER_TAG
	ITERATOR	868	2		:Q1000	PCWC
	FULL	868	2		:Q1000	PCWP
		1000	2		:Q1002	PCWP
	HYBRID HASH	1000	2	HYBRID HASH	:Q1001	P->P
	ITERATOR	1000	2		:Q1001	PCWC
	FULL	1000	2		:Q1001	PCWP

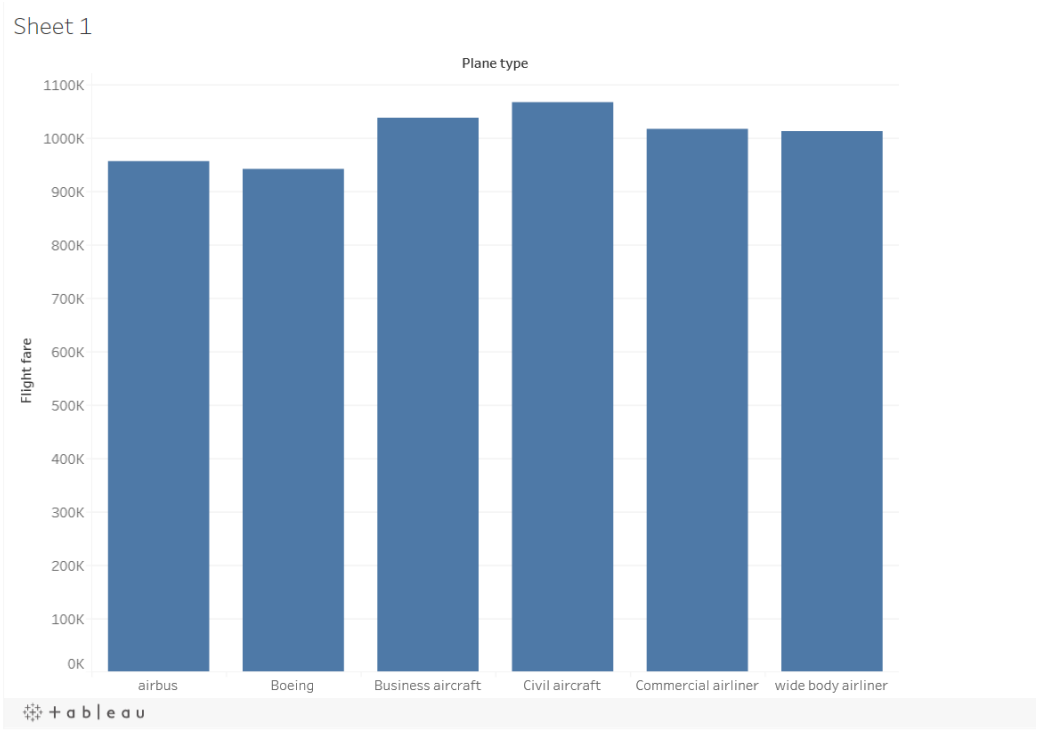
Here it comes out that the cost of execution for a single operation has further decreased to 2 and Hence, Parallel Processing as helped us in Performance Tuning of our database system.

8. Data Analysis with Tableau

We have used Tableau for analysis of some scenario which can help in decision like determining which Airport has longest wait time and how much is total fare according to plane type.

1) Total flight fare for flight with plane type

   For determining total Flight fare for flight for particular plane type we have used two tables Plane and Booking. We have used full outer join on Plane and Booking on Flight_Number. In order Total Fare according to particular Plane type we have selected Plane type as column and Sum of Flight Fare as row. In order to display graphically we have used graph chart as below.



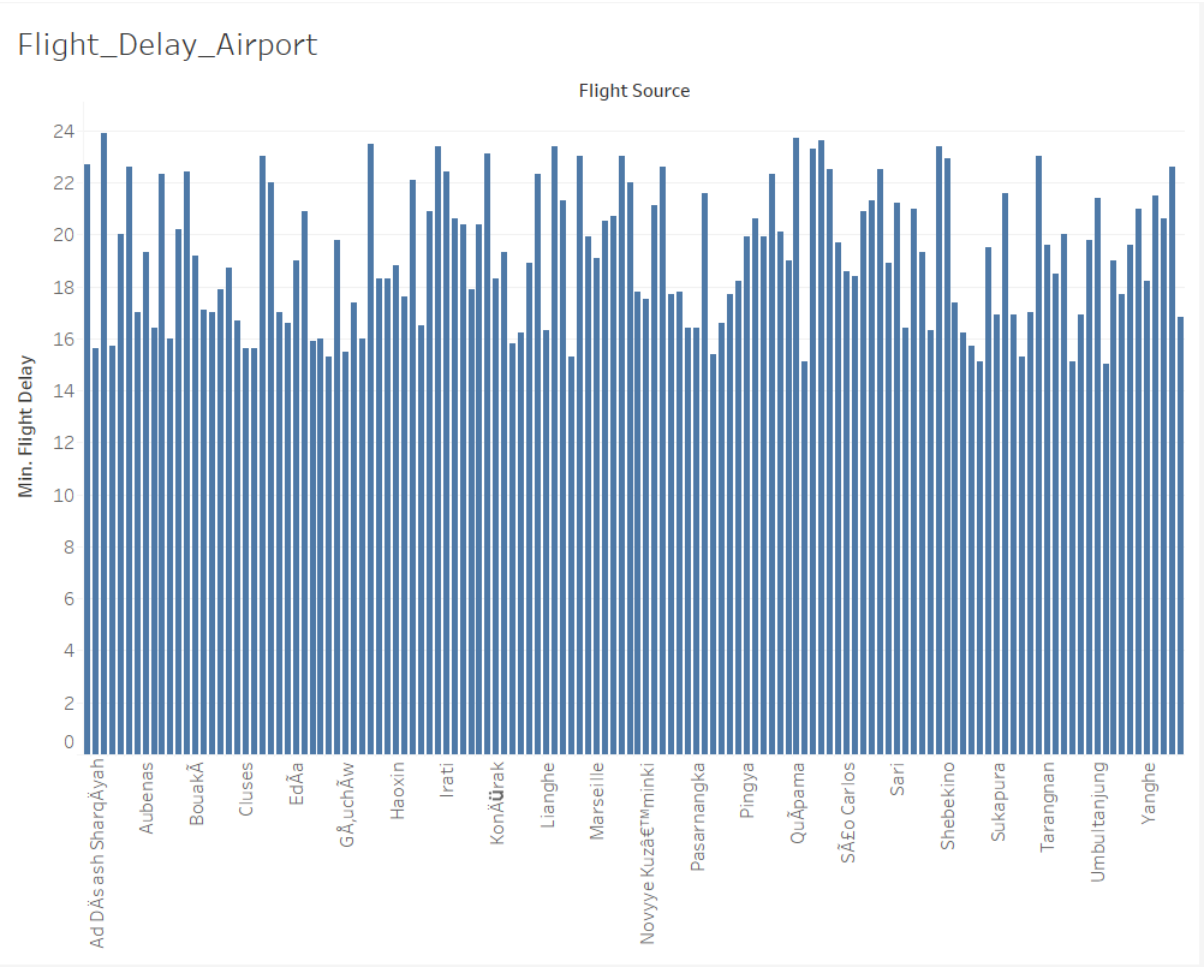
To understand total Fare in chart with numerical values as below we have selected text tabular chart for Fare details.

Flight_Fare_Total_Plane_Type

Plane type	
airbus	957,158
Boeing	941,614
Business aircraft	1,037,764
Civil aircraft	1,067,380
Commercial airliner	1,016,139
wide body airliner	1,011,973

2) Airports with delay greater than 15 hours

In order to determining Airports with Flight delay for flight for particular plane type we have used two tables Plane and CSV. We have used full outer join on Plane and Flight on Flight_Number. Flight_Source has Airport City and it is set as column. Flight_Delay has delay in Flight departure in hours from Airport. Flight_Delay is set as Row with Minimum delay as 15 hours. We will get plot as below in Bar chart.


Flight_Delay_Airport
Flight Source
Min. Flight Delay
24
22
20
18
16
14
12
10
8
6
4
2
0
Ad DÃs ash SharqÃyah
Aubenas
BouakÃ
Cluses
EdÅa
GÅ¸uchÃw
Haoxin
Irati
KonÄürak
Lianghe
Marseille
Novyye Kuzâ€™minki
Pasarnangka
Pingya
QuÃ¸pama
SÃ£o Carlos
Sari
Shebekino
Sukapura
Tarangnan
Umbultanjung
Yanghe

9. Key Results and Discussion

We have successfully created database for Airline Management System by generating data and import same in Oracle SQL developer. After data is imported in Database Primay key and Foreign keys were defined for removing data redundancy. We have built some queries for various scenarios and wrote plsql procedure. We have also done performance optimization. As it comes out that after introducing parallel processing into our database system for execution of our cost of execution gets reduced significantly. This happened because we assigned parallel servers for each of the single operation and thus reducing the overall time of execution and thus expediting the process. We have also done some Data Analysis using Tableau and got interesting charts with same.