Airline case study: data transformation

José Luis Puente Bodoque

January, 2023

Abstract

Pentaho Data Integration is for applying a ETL process on data input in order to populate a data warehouse. In this report we describe in depth the logic behind each of those processes on the airline case study.

Contents

1 Building dimension tables

The table 1 shows a reasonable correspondence between the data source and dimension tables.

Input file name	Dimension table
`airport.csv`	airport
`airport_city_state.csv`	
`fare.csv`	fare
`channel.csv`	payment_channel
`customer.csv`	passenger
`flight.csv`	airplane
`hour.csv`	hour

Table 1: Correspondence between input and dimension

Below we step into the explanation of each ETL process.

1.1 Airport

The figure 1 shows the transformation performed to build the airport dimension table.

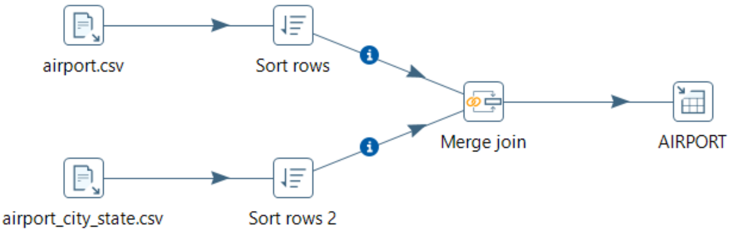
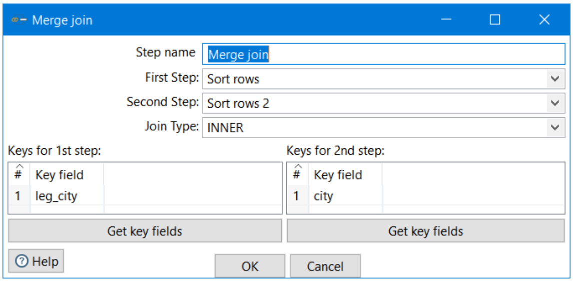


Figure 1: ETL structure of airport dimension

Taking a look `airport.csv` we notice state field is missing although we does find in `airport_city_state.csv`. In order to fetch state field to the main stream we use **Merge join** step with data coming from the two CSV input files. We set join type as INNER so that only rows having the same *city* keys in both sources be included in the result (figure 2). If we are careful to read the web documentation on Merge Join step we will run into a note that says: "Input rows are



Step name: Merge join
First Step: Sort rows
Second Step: Sort rows 2
Join Type: INNER

Keys for 1st step:

#	Key field
1	leg_city

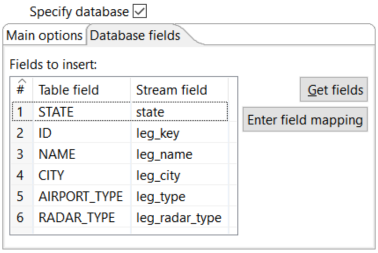
Keys for 2nd step:

#	Key field
1	city

Figure 2: Merge Join step

expected to be sorted on the specified key fields”. Thus we have to include one **Sort Row** step before each Merge Join step.

The figure 3 shows the field mapping between stream fields and table fields.



Specify database ☑

Fields to insert:

#	Table field	Stream field
1	STATE	state
2	ID	leg_key
3	NAME	leg_name
4	CITY	leg_city
5	AIRPORT_TYPE	leg_type
6	RADAR_TYPE	leg_radar_type

Figure 3: Field mapping of airport dimension

1.2 Fare

The figure 4 shows the ETL process performed to build the fare dimension table. It consists simply of the extraction and data load of input data into the output table.

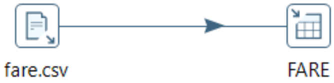
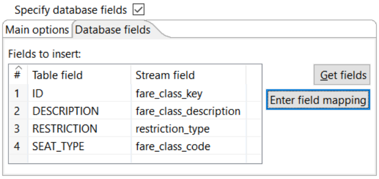


Figure 4: ETL strcture of fare dimension

The figure 5 shows the field mapping between stream fields and table fields.



Specify database fields ☑

Fields to insert:

#	Table field	Stream field
1	ID	fare_class_key
2	DESCRIPTION	fare_class_description
3	RESTRICTION	restriction_type
4	SEAT_TYPE	fare_class_code

Figure 5: Field mapping of fare dimension

1.3 Hour

The figure 6 shows the ETL process performed to build the hour dimension table. It consists simply of the extraction and data load of input data into the output table..

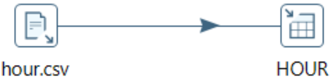
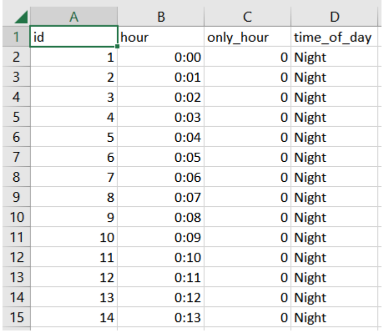


Figure 6: ETL structure of hour dimension

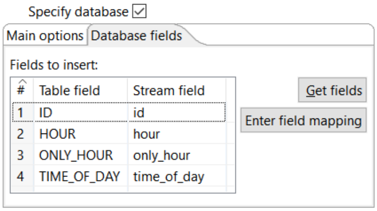
`hour.csv` is a handmade file to save all the possible combinations of hour and moment of day. It is not provided but it's essential for finding arrival and departure time identifiers later. The figure 7 shows the header and the first 15 rows as sample.



	A	B	C	D
1	id	hour	only_hour	time_of_day
2	1	0:00	0	Night
3	2	0:01	0	Night
4	3	0:02	0	Night
5	4	0:03	0	Night
6	5	0:04	0	Night
7	6	0:05	0	Night
8	7	0:06	0	Night
9	8	0:07	0	Night
10	9	0:08	0	Night
11	10	0:09	0	Night
12	11	0:10	0	Night
13	12	0:11	0	Night
14	13	0:12	0	Night
15	14	0:13	0	Night

Figure 7: Sample of `hour.csv`

The figure 8 shows the field mapping between stream fields and table fields.



Fields to insert:

#	Table field	Stream field
1	ID	id
2	HOUR	hour
3	ONLY_HOUR	only_hour
4	TIME_OF_DAY	time_of_day

Figure 8: Field mapping of hour dimension

1.4 Payment Channel

The figure 9 shows the ETL process performed to build the payment channel dimension table. It consists simply of the extraction and data load of input data into the output table.

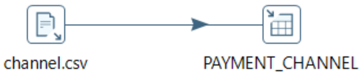
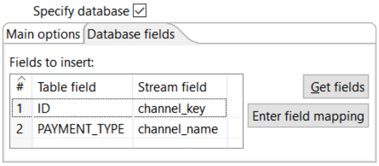


Figure 9: ETL structure for payment channel dimension

The figure 10 shows the field mapping between stream fields and table fields.



Fields to insert:

#	Table field	Stream field
1	ID	channel_key
2	PAYMENT_TYPE	channel_name

Figure 10: Field mapping of payment channel dimension

1.5 Passenger

The figure 11 shows the ETL process performed to build the passenger dimension table. It consists simply of the extraction and data load of input data into the output table.

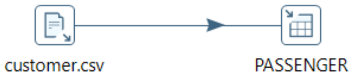
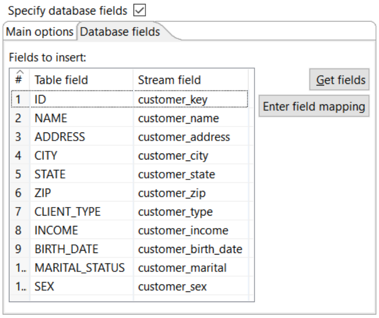


Figure 11: ETL structure of passenger dimension

The figure 12 shows the field mapping between stream fields and table fields.



Specify database fields ☑

Fields to insert:

#	Table field	Stream field
1	ID	customer_key
2	NAME	customer_name
3	ADDRESS	customer_address
4	CITY	customer_city
5	STATE	customer_state
6	ZIP	customer_zip
7	CLIENT_TYPE	customer_type
8	INCOME	customer_income
9	BIRTH_DATE	customer_birth_date
1..	MARITAL_STATUS	customer_marital
1..	SEX	customer_sex

Figure 12: Field mapping of passenger dimension

1.6 Airplane

The figure 13 shows the ETL process performed to build the airplane dimension table. It consists simply of the extraction and data load of input data into the output table.

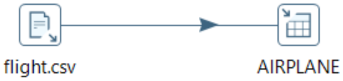
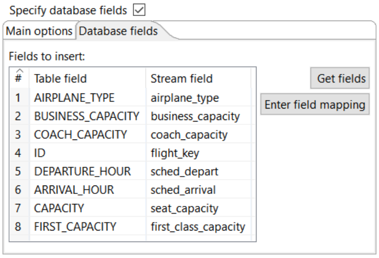


Figure 13: ETL structure of airplane dimension

The figure 14 shows the field mapping between stream fields and table fields.



Specify database fields ☑

Fields to insert:

#	Table field	Stream field
1	AIRPLANE_TYPE	airplane_type
2	BUSINESS_CAPACITY	business_capacity
3	COACH_CAPACITY	coach_capacity
4	ID	flight_key
5	DEPARTURE_HOUR	sched_depart
6	ARRIVAL_HOUR	sched_arrival
7	CAPACITY	seat_capacity
8	FIRST_CAPACITY	first_class_capacity

Figure 14: Field mapping of airplane dimension

2 Building the fact table

`frequentflyer.csv` file aims to be the input of the fact table. However there's an issue with the file content that in fact we talked about in the first report.

2.1 Issue with the input

`frequentflyer.csv` lacks the departure and arrival hour keys. The keys to look for are either in `hour.csv` or in the hour dimension table. This drives us to evaluate two possible solutions for loading the data.

2.2 ETL load process

The first option to load the data consists of using the **Database Lookup** step (figure 15) and the second one, the **Stream Lookup** one (figure 16). Both works fine but are quite slow. The structure presents a neck bottle in the last output table step that slows down the pace of implementation and delays the process up to 5 minutes. The figure 17 shows the metrics of the transformation. May the first option is more convenient in business environments because the keys we need are always in a certain dimension table and not necessarily in the data source.

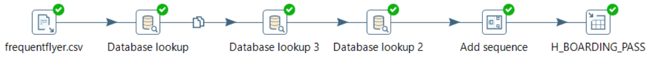


Figure 15: First option

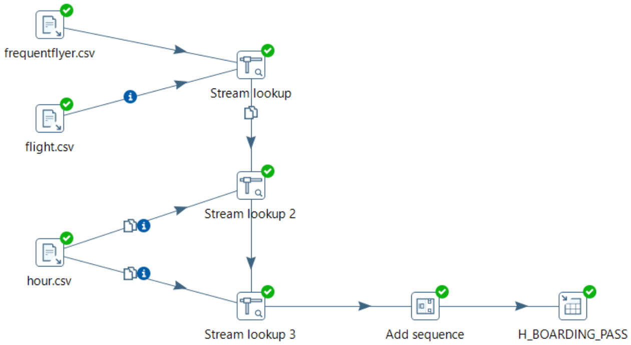
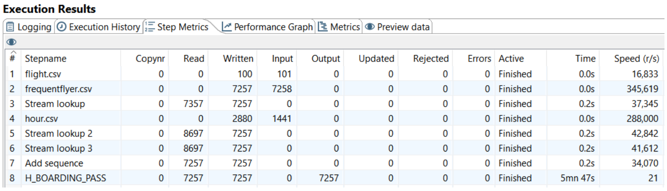


Figure 16: Second option

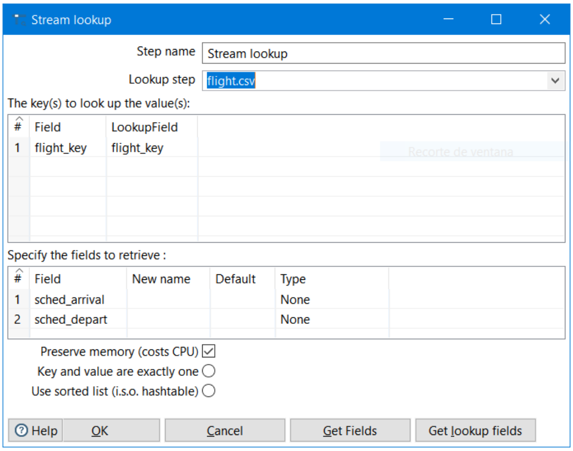


Execution Results

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)
1	flight.csv	0	0	100	101	0	0	0	0	Finished	0.0s	16,833
2	frequentflyer.csv	0	0	7257	7258	0	0	0	0	Finished	0.0s	345,619
3	Stream lookup	0	7357	7257	0	0	0	0	0	Finished	0.2s	37,345
4	hour.csv	0	0	2880	1441	0	0	0	0	Finished	0.0s	288,000
5	Stream lookup 2	0	8697	7257	0	0	0	0	0	Finished	0.2s	42,842
6	Stream lookup 3	0	8697	7257	0	0	0	0	0	Finished	0.2s	41,612
7	Add sequence	0	7257	7257	0	0	0	0	0	Finished	0.2s	34,070
8	H_BOARDING_PASS	0	7257	7257	0	7257	0	0	0	Finished	5mn 47s	21

Figure 17: Step metrics for fact table

The figure 18 shows how we have looked up departure and arrival hours on flight keys.



Stream lookup

Step name: Stream lookup

Lookup step: flight.csv

The key(s) to look up the value(s):

#	Field	LookupField
1	flight_key	flight_key

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	sched_arrival			None
2	sched_depart			None

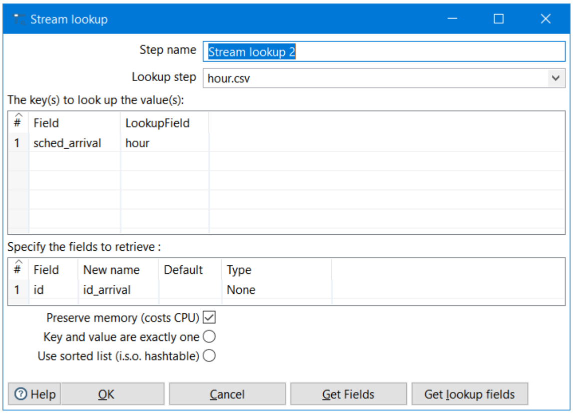
Preserve memory (costs CPU) ☑

Key and value are exactly one ○

Use sorted list (i.s.o. hashtable) ○

Figure 18: First Stream lookup step

The figure 19 shows how we have looked up the id_arrival key on hour key and sched_arrival key.



Step name: Stream lookup 2

Lookup step: hour.csv

The key(s) to look up the value(s):

#	Field	LookupField
1	sched_arrival	hour

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	id	id_arrival		None

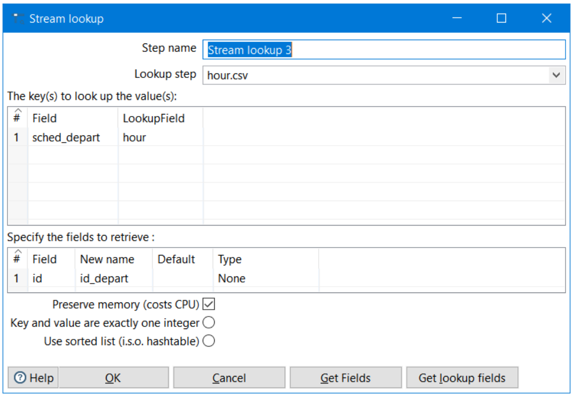
Preserve memory (costs CPU) ☑

Key and value are exactly one ○

Use sorted list (i.s.o. hashtable) ○

Figure 19: Second Stream lookup 2

The figure 20 shows how we have looked up the id_depart key on both hour key and sched_depart key.



Step name: Stream lookup 3

Lookup step: hour.csv

The key(s) to look up the value(s):

#	Field	LookupField
1	sched_depart	hour

Specify the fields to retrieve :

#	Field	New name	Default	Type
1	id	id_depart		None

Preserve memory (costs CPU) ☑

Key and value are exactly one integer ○

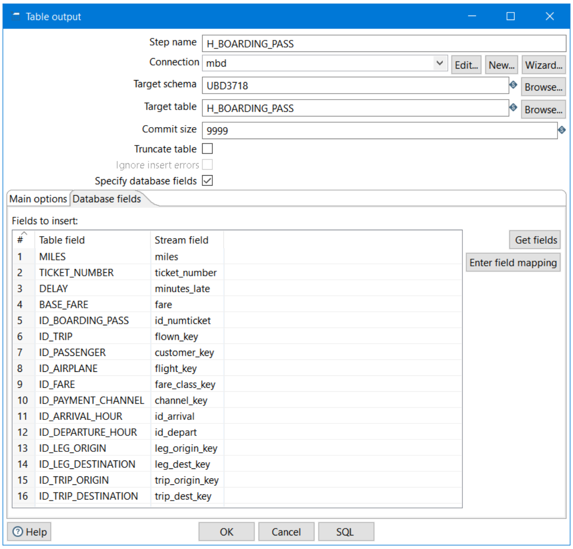
Use sorted list (i.s.o. hashtable) ○

Figure 20: Third Stream lookup 3

Add Sequence step generates an incremental sequence of integer values for the surrogate key of the fact table..

Finally the figure 21 shows the mapping between the fields coming from the main stream and fact table fields.

Note: the input table has 7257 rows so we must change the commit size to 7257 or greater to avoid an error.



Step name: H_BOARDING_PASS

Connection: mbd

Target schema: UBD3718

Target table: H_BOARDING_PASS

Commit size: 9999

Truncate table ☐

Ignore insert errors ☐

Specify database fields ☑

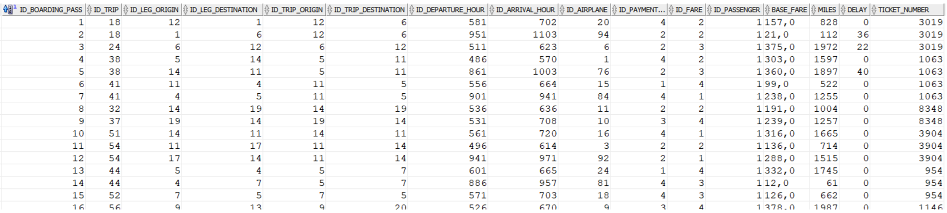
Fields to insert:

#	Table field	Stream field
1	MILES	miles
2	TICKET_NUMBER	ticket_number
3	DELAY	minutes_late
4	BASE_FARE	fare
5	ID_BOARDING_PASS	id_numticket
6	ID_TRIP	flown_key
7	ID_PASSENGER	customer_key
8	ID_AIRPLANE	flight_key
9	ID_FARE	fare_class_key
10	ID_PAYMENT_CHANNEL	channel_key
11	ID_ARRIVAL_HOUR	id_arrival
12	ID_DEPARTURE_HOUR	id_depart
13	ID_LEG_ORIGIN	leg_origin_key
14	ID_LEG_DESTINATION	leg_dest_key
15	ID_TRIP_ORIGIN	trip_origin_key
16	ID_TRIP_DESTINATION	trip_dest_key

Figure 21: Field mapping of fact table

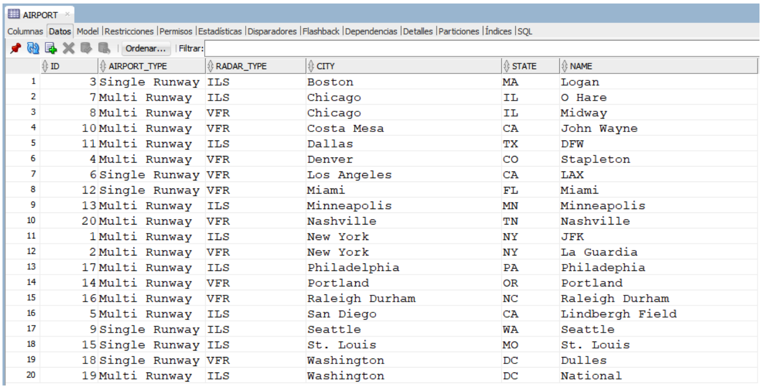
3 Ailine data warehouse

This is how our data warehouse for the airline case study looks like.



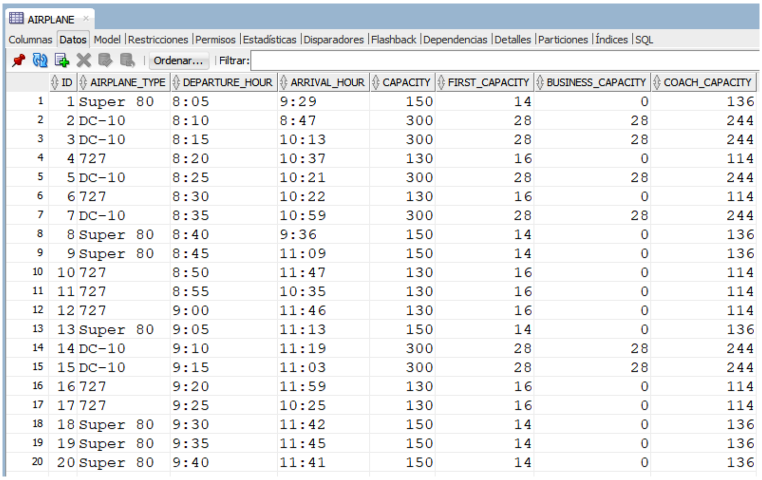
ID_BOARDING_PASS	ID_TRIP	ID_LEG_ORIGIN	ID_LEG_DESTINATION	ID_TRIP_ORIGIN	ID_TRIP_DESTINATION	ID_DEPARTURE_HOUR	ID_ARRIVAL_HOUR	ID_AIRPLANE	ID_PAYMENT...	ID_FARE	ID_PASSENGER	BASE_FARE	MILES	DELAY	TICKET_NUMBER
1	18	12	1	12	6	581	702	20	4	2	1	157,0	828	0	3019
2	18	1	6	12	6	951	1103	94	2	2	1	21,0	112	36	3019
3	24	6	12	6	12	511	623	6	2	3	1	375,0	1972	22	3019
4	38	5	14	5	11	486	570	1	4	2	1	303,0	1597	0	1063
5	38	14	11	5	11	861	1003	76	2	3	1	360,0	1897	40	1063
6	41	11	4	11	5	556	664	15	1	4	1	99,0	522	0	1063
7	41	4	5	11	5	901	941	84	4	1	1	238,0	1255	0	1063
8	32	14	19	14	19	536	636	11	2	2	1	191,0	1004	0	8348
9	37	19	14	19	14	531	708	10	3	4	1	239,0	1257	0	8348
10	51	14	11	14	11	561	720	16	4	1	1	316,0	1665	0	3904
11	54	11	17	11	14	496	614	3	2	2	1	136,0	714	0	3904
12	54	17	14	11	14	941	971	92	2	1	1	288,0	1515	0	3904
13	44	5	4	5	7	601	665	24	1	4	1	332,0	1745	0	954
14	44	4	7	5	7	886	957	81	4	3	1	12,0	61	0	954
15	52	7	5	7	5	571	703	18	4	3	1	126,0	662	0	954
16	56	9	13	9	20	526	670	9	3	4	1	378,0	1987	0	1146

Figure 22: Fact table


AIRPORT
Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL
Ordenar...
Filtrar:

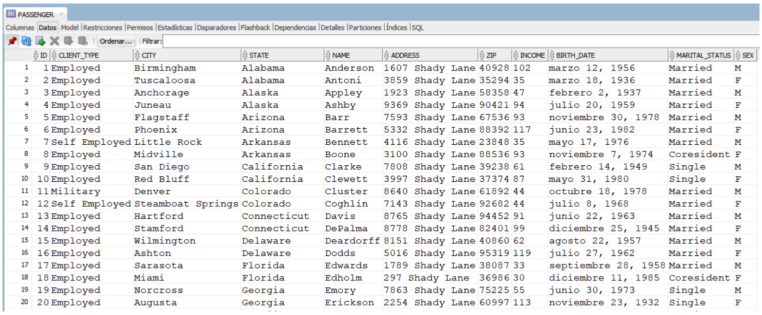
	ID	AIRPORT_TYPE	RADAR_TYPE	CITY	STATE	NAME
1	3	Single Runway	ILS	Boston	MA	Logan
2	7	Multi Runway	ILS	Chicago	IL	O Hare
3	8	Multi Runway	VFR	Chicago	IL	Midway
4	10	Multi Runway	VFR	Costa Mesa	CA	John Wayne
5	11	Multi Runway	ILS	Dallas	TX	DFW
6	4	Multi Runway	VFR	Denver	CO	Stapleton
7	6	Single Runway	VFR	Los Angeles	CA	LAX
8	12	Single Runway	VFR	Miami	FL	Miami
9	13	Multi Runway	ILS	Minneapolis	MN	Minneapolis
10	20	Multi Runway	VFR	Nashville	TN	Nashville
11	1	Multi Runway	ILS	New York	NY	JFK
12	2	Multi Runway	VFR	New York	NY	La Guardia
13	17	Multi Runway	ILS	Philadelphia	PA	Philadephia
14	14	Multi Runway	VFR	Portland	OR	Portland
15	16	Multi Runway	VFR	Raleigh Durham	NC	Raleigh Durham
16	5	Multi Runway	ILS	San Diego	CA	Lindbergh Field
17	9	Single Runway	ILS	Seattle	WA	Seattle
18	15	Single Runway	ILS	St. Louis	MO	St. Louis
19	18	Single Runway	VFR	Washington	DC	Dulles
20	19	Multi Runway	ILS	Washington	DC	National

Figure 23: Airport dimension table


AIRPLANE
Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL
Ordenar...
Filtrar:

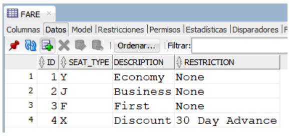
	ID	AIRPLANE_TYPE	DEPARTURE_HOUR	ARRIVAL_HOUR	CAPACITY	FIRST_CAPACITY	BUSINESS_CAPACITY	COACH_CAPACITY
1	1	Super 80	8:05	9:29	150	14	0	136
2	2	DC-10	8:10	8:47	300	28	28	244
3	3	DC-10	8:15	10:13	300	28	28	244
4	4	727	8:20	10:37	130	16	0	114
5	5	DC-10	8:25	10:21	300	28	28	244
6	6	727	8:30	10:22	130	16	0	114
7	7	DC-10	8:35	10:59	300	28	28	244
8	8	Super 80	8:40	9:36	150	14	0	136
9	9	Super 80	8:45	11:09	150	14	0	136
10	10	727	8:50	11:47	130	16	0	114
11	11	727	8:55	10:35	130	16	0	114
12	12	727	9:00	11:46	130	16	0	114
13	13	Super 80	9:05	11:13	150	14	0	136
14	14	DC-10	9:10	11:19	300	28	28	244
15	15	DC-10	9:15	11:03	300	28	28	244
16	16	727	9:20	11:59	130	16	0	114
17	17	727	9:25	10:25	130	16	0	114
18	18	Super 80	9:30	11:42	150	14	0	136
19	19	Super 80	9:35	11:45	150	14	0	136
20	20	Super 80	9:40	11:41	150	14	0	136

Figure 24: Airplane dimension table


PASSENGER
Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL
Ordenar...
Filtrar:

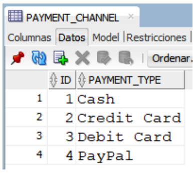
	ID	CLIENT_TYPE	CITY	STATE	NAME	ADDRESS	ZIP	INCOME	BIRTH_DATE	MARITAL_STATUS	SEX
1	1	Employed	Birmingham	Alabama	Anderson	1607 Shady Lane	40928	102	marzo 12, 1956	Married	M
2	2	Employed	Tuscaloosa	Alabama	Antoni	3859 Shady Lane	35294	35	marzo 18, 1936	Married	F
3	3	Employed	Anchorage	Alaska	Appley	1923 Shady Lane	58358	47	febrero 2, 1937	Married	M
4	4	Employed	Juneau	Alaska	Ashby	9369 Shady Lane	90421	94	julio 20, 1959	Married	F
5	5	Employed	Flagstaff	Arizona	Barr	7593 Shady Lane	67536	93	noviembre 30, 1978	Married	M
6	6	Employed	Phoenix	Arizona	Barrett	5332 Shady Lane	88392	117	junio 23, 1982	Married	F
7	7	Self Employed	Little Rock	Arkansas	Bennett	4116 Shady Lane	23848	35	mayo 17, 1976	Married	M
8	8	Employed	Midville	Arkansas	Boone	3100 Shady Lane	88536	93	noviembre 7, 1974	Coresident	F
9	9	Employed	San Diego	California	Clarke	7808 Shady Lane	39238	61	febrero 14, 1949	Single	M
10	10	Employed	Red Bluff	California	Clewett	3997 Shady Lane	37374	87	mayo 31, 1980	Single	F
11	11	Military	Denver	Colorado	Cluster	8640 Shady Lane	61892	44	octubre 18, 1978	Married	M
12	12	Self Employed	Steamboat Springs	Colorado	Coghlin	7143 Shady Lane	92682	44	julio 8, 1968	Married	F
13	13	Employed	Hartford	Connecticut	Davis	8765 Shady Lane	94452	91	junio 22, 1963	Married	M
14	14	Employed	Stamford	Connecticut	DePalma	8778 Shady Lane	82401	99	diciembre 25, 1945	Married	F
15	15	Employed	Wilmington	Delaware	Deardorff	8151 Shady Lane	40860	62	agosto 22, 1957	Married	M
16	16	Employed	Ashton	Delaware	Dodds	5016 Shady Lane	95319	119	julio 27, 1962	Married	F
17	17	Employed	Sarasota	Florida	Edwards	1789 Shady Lane	38087	33	septiembre 28, 1958	Married	M
18	18	Employed	Miami	Florida	Edholm	297 Shady Lane	36986	30	diciembre 11, 1985	Coresident	F
19	19	Employed	Norcross	Georgia	Emory	7863 Shady Lane	75225	55	junio 30, 1973	Single	M
20	20	Employed	Augusta	Georgia	Erickson	2254 Shady Lane	60997	113	noviembre 23, 1932	Single	F

Figure 25: Passenger dimension table



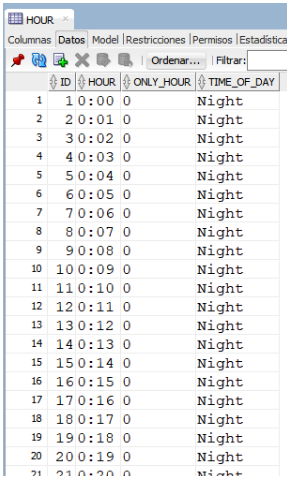
	ID	SEAT_TYPE	DESCRIPTION	RESTRICTION
1	1	Y	Economy	None
2	2	J	Business	None
3	3	F	First	None
4	4	X	Discount	30 Day Advance

Figure 26: Fare dimension table



	ID	PAYMENT_TYPE
1	1	Cash
2	2	Credit Card
3	3	Debit Card
4	4	PayPal

Figure 27: Payment Channel dimension table



	ID	HOUR	ONLY_HOUR	TIME_OF_DAY
1	1	0:00	0	Night
2	2	0:01	0	Night
3	3	0:02	0	Night
4	4	0:03	0	Night
5	5	0:04	0	Night
6	6	0:05	0	Night
7	7	0:06	0	Night
8	8	0:07	0	Night
9	9	0:08	0	Night
10	10	0:09	0	Night
11	11	0:10	0	Night
12	12	0:11	0	Night
13	13	0:12	0	Night
14	14	0:13	0	Night
15	15	0:14	0	Night
16	16	0:15	0	Night
17	17	0:16	0	Night
18	18	0:17	0	Night
19	19	0:18	0	Night
20	20	0:19	0	Night
21	21	0:20	0	Night

Figure 28: Hour dimension table