DATA WAREHOUSE PROJECT

-Ruchika Dhungana

Table of Contents

Project Introduction

The project is based off an airline company named FlyU. They hold records related to their flights such as: departure times, arrival times, customers on each flight, number of flights to reach their destination, return flights, tail number, complaints raised, and any type of flight delays.

The company stores using Oracle as well as Excel sheets. They handle these day to day data in order to plan, manage their customer complaints service and deliver a quality service.

Aims/Objectives

FlyU company aims for the following results:

- Integrate a Data Warehouse to store the information relating to FlyU.
- Deliver a quality service
- Increase customer satisfaction
- Grow the company.

Data Warehouse

Before we move further into the project, it is crucial to understand what data warehouse is. So, in simple terms, it is a type of data management system that stores a company's data from one or more sources. The main purpose of data warehouse is to compare and analyse data for greater corporate performance. When it comes to business intelligence, it is considered one the vital components as it helps support business decisions by providing analytical techniques and an overall broader insight into the performance of a company. It is designed to execute query and analysis on historical data derived from heterogeneous sources.

Main Objective:

These are the main objective of implementing a Data Warehouse into a company:

- **Improve quality of data**:

  One of the main purpose of Data Warehouse is to guarantee data quality. Any bad or error data are analysed, purified, and transformed into a useable data hence ensuring good data quality.

- **Minimize inconsistent reports:**

  Since, inconsistent reports are mainly caused by misuse of data, and the main reason for misuse of data is disagreement or misunderstanding of the meaning or the content of data. Data Warehouse ensures that there is no disagreements or misunderstandings.

- **Integrate data from multiple sources:**

  Another prime objective of Data Warehouse is to make it easy for companies to integrate data from multiple sources.

- **Merge historical data with current data:**

  As source systems do not usually keep a history of certain data, typical data warehouse objective is to store history. In data warehouse data changes in the source system are recorded, which enables historical analysis.

OLAP:

OLAP (*online analytical processing*) is a core component of data warehouse which enables multidimensional data analysis for business intelligence (BI) and greater corporate performance.

OLAP cubes enable four basic types of multidimensional data analysis:

- **Drill-down:** The drill-down operation converts less-detailed data into more-detailed data
- **Roll-up:** Roll up is the opposite of the drill-down function as it aggregates detailed data.
- **Slice and Dice:** The slice operation creates a sub-cube by selecting a single dimension and the dice operation isolates a sub-cube by selecting several dimensions.
- **Pivot:**The pivot function rotates the current cube view to enable dynamic multidimensional views of data.

OLTP:

OLTP(*online transaction processing*) enables transaction-oriented applications in a 3-tier architecture. OLTP administers day to day transaction of an organization.

OLAP Vs OLTP:

The differences between OLAP and OLTP:

OLAP	OLTP
The primary objective is data analysis.	The primary objective is data processing.
OLAP can be used for all type of business analysis needs which	OLTP is useful to administer day to

includes planning, budgeting, forecasting, and analysis	day transactions of an organization.
OLAP uses data warehouse technique where it can integrate different data sources for building a secure database.	OLTP uses traditional DBMS

KPI 2: Ensure customer satisfaction

As FlyU company is looking to integrate a Data Warehouse to focus on their **Key Performance Indicators (KPI)** which includes deliver a quality service, increase customer satisfaction and grow the company. We will be focusing on the second KPI i.e. ensure customer satisfaction.

This KPI will focus on all the data that are necessary to make sure that the company will be able to reach a decision on how to increase their customer satisfaction, based on the analysis report that we will create using OLAP system.

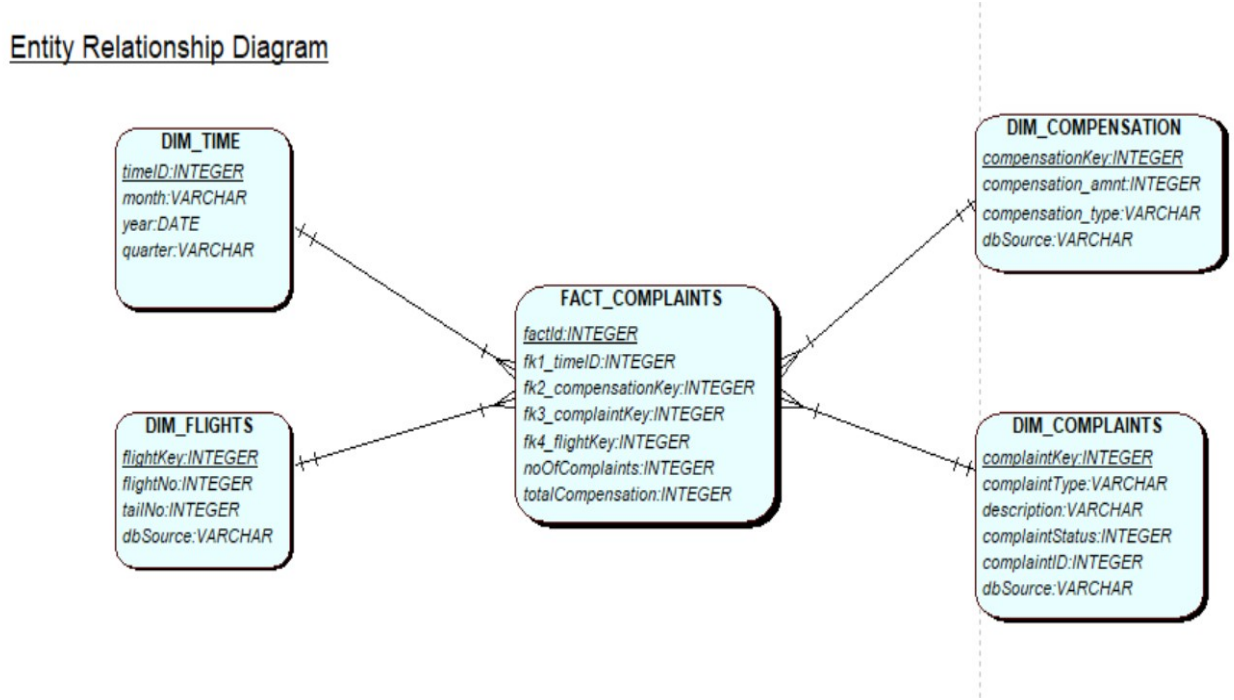
Task 1

REPORTS:

These are the reports that will be generated once the project is completed.

- Identify the airlines with most complaints.
- The no. of complaints per flight.
- Total amount of compensation given each month in the year 2017.
- The no. of complain per flight per month in the year 2017.
- The total number of compensation given in 2017, according to the type of complaints.

STAR SCHEMA:



DATA DICTIONARY:

Data dictionary provides a clear picture about the contents, format, and structure of a database and the relationship between its elements as well as any bad data that need to be transformed.

- DIM_TIME

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
DIM_TIME	timeID	Integer	Yes	Automatically generated as a primary key	n/a	n/a	Create a sequence timeid_seq to generate primary keys
	year	Number	No	FlyU_flights	n/a	n/a	n/a
	month	Number	No	FlyU_flights	n/a	n/a	n/a
	day	Number	No	FlyU_flights	n/a	n/a	n/a
Definition	The dim_time table holds the intervals of time for which the data will be held.						

:	It is held at year, month, day and quarterly level meaning.
Notes:	

- DIM_FLIGHTS

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
DIM_FLIGHTS	flightKey	INTEGER	Yes	Automatically generated as surrogate key	n/a	n/a	Create a sequence flight_seq to generate primary keyS
	flightNo	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	TailNo	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	dbSource	VARCHAR	No	Should be generated	n/a	n/a	Create a sequence SOURCE_seq to generate Quarterly dates
Definition:	The dim_FLIGHTS table holds the data related to all the flights.						
Notes:							

- DIM_COMPENSATION

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
DIM_COMPENSATION	compensationKey	INTEGER	Yes	Automatically generated as surrogate key	n/a	n/a	Create a sequence comp_seq to generate primary keys
	compensation_amnt	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	compensation_type	VARCH	No	FlyU_flights	Null value	Some complaints	Will need to

		AR				are missing the compensation type	mention the compensation type
					Inconsistent Data	Inconsistency in compensation type. Eg: ‘rebooked’, ‘rebook’.	Will need to convert the compensation type from ‘rebook’ to rebooked.
	dbSource	VARCHAR	No	Should be generated	n/a	n/a	Create a sequence SOURCE_seq to generate quarterly dates
Definition:	The dim_compensation table holds the data related to the compensation given to the customers.						
Notes:							

- DIM_COMPLAINTS

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
DIM_ COMPLAINTS	Complaint Key	INTEGER	Yes	Automatically generated as surrogate key	n/a	n/a	Create a sequence complain_seq to generate primary keys
	Complaint Id	INTEGER	No	FlyU_flights	n/a	n/a	n/a

	Complaint Type	VARCHAR	No	FlyU_flights	Null value	Some complaints are missing the compensation type	Will need to add the missing compensation type
					Inconsistent value	Complaint types have irregural values. Eg: A,B,C for cancellation	Will need to transform all the irregular values to ‘C’ for cancellation and ‘L’ for late.
	Description	VARCHAR	No	FlyU_flights	Null Value	Some complaints are missing the description	Will need to add the missing description
	Complaint status	VARCHAR	No	FlyU_flights	n/a	n/a	n/a
	dbSource	VARCHAR	No	Should be generated	n/a	n/a	Create a sequence SOURCE_seq to generate quarterly dates
Definition:	The dim_complaint table holds all the data related to the customer complaints.						
Notes:							

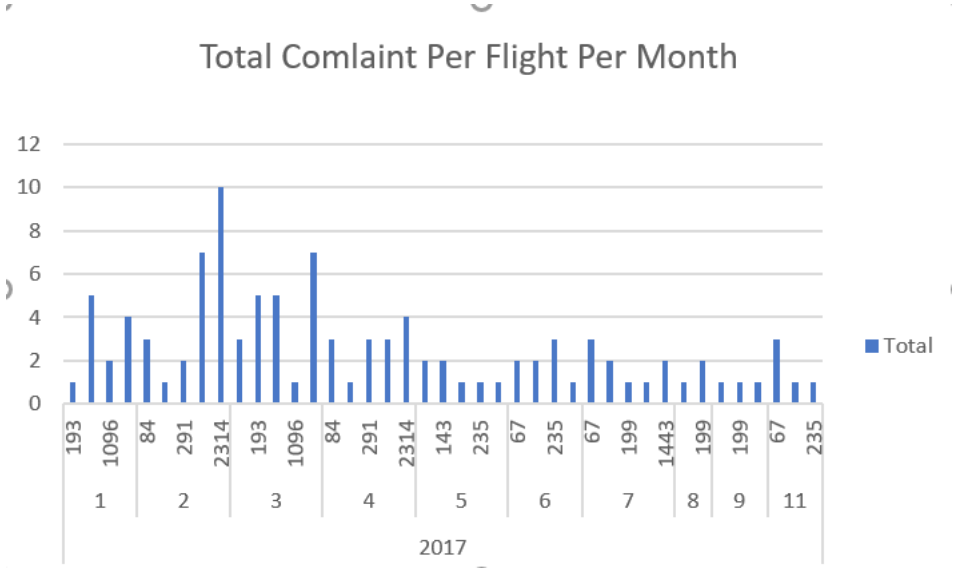
- FACT_COMPLAINTS

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
FACT_ COMPLAIN	FactId	INTEGER	Yes	Automatically generate	n/a	n/a	Create a sequence

				d as primary key			fact_seq to generate primary keys
	Complaint Key	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	Compensation Key	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	TimeId	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	flightKey	INTEGER	No	FlyU_flights	n/a	n/a	n/a
TS	noOfComplaint	INTEGER	No	Should be generated	n/a	n/a	Create a sequence SOURCE_seq to generate quarterly dates
	total Compensation	INTEGER	No	Should be generated	n/a	n/a	Create a sequence SOURCE_seq to generate quarterly dates
Definition:	The dim_complaint table holdsall the data related to the customer complaints.						
Notes:							

REPORT:

- EXCEL REPORT

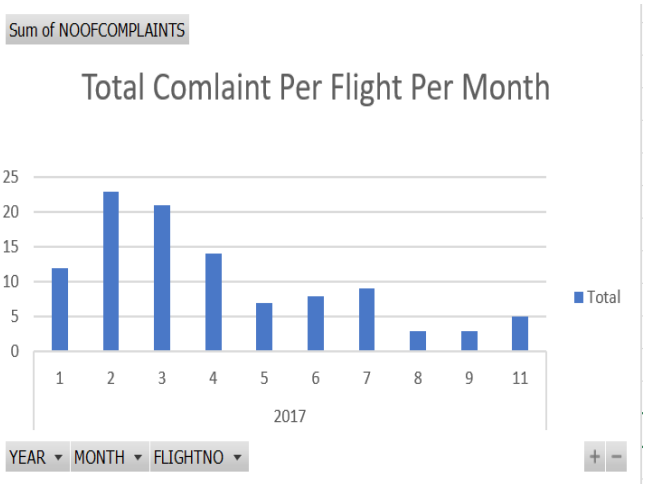


- Drill Down Report

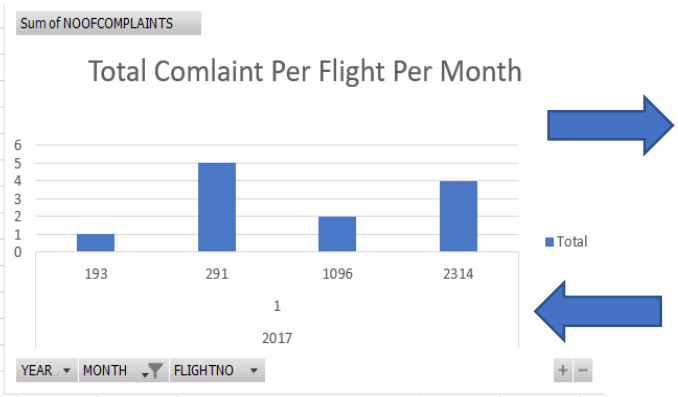
YEAR	MONTH	FLIGHTNO	Sum of NOOFCOMPLAINTS
2017	**1**	193	1
		291	5
		1096	2
		2314	4
	1 Total		**12**
	2	84	3
		193	1
		291	2
		1096	7
		2314	10
	2 Total		**23**
	3	84	3
		193	5
		291	5
		1096	1
		2314	7
	3 Total		**21**

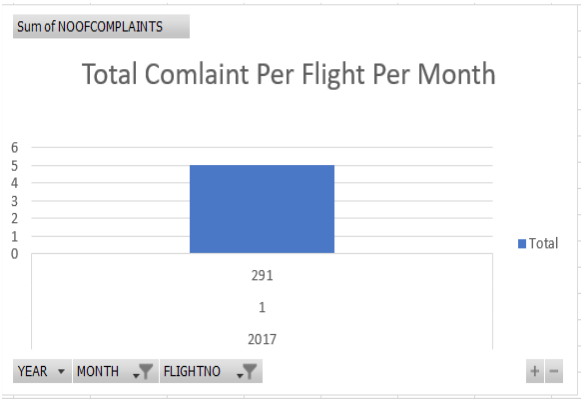
- Rollup Report

YEAR	MONTH	FLIGHTNO	Sum of NOOFCOMPLAINTS
2017	1		12
	2		23
	3		21
	4		14
	5		7
	6		8
	7		9
	8		3
	9		3
	11		5
2017 Total			105



- SLICING/DICING



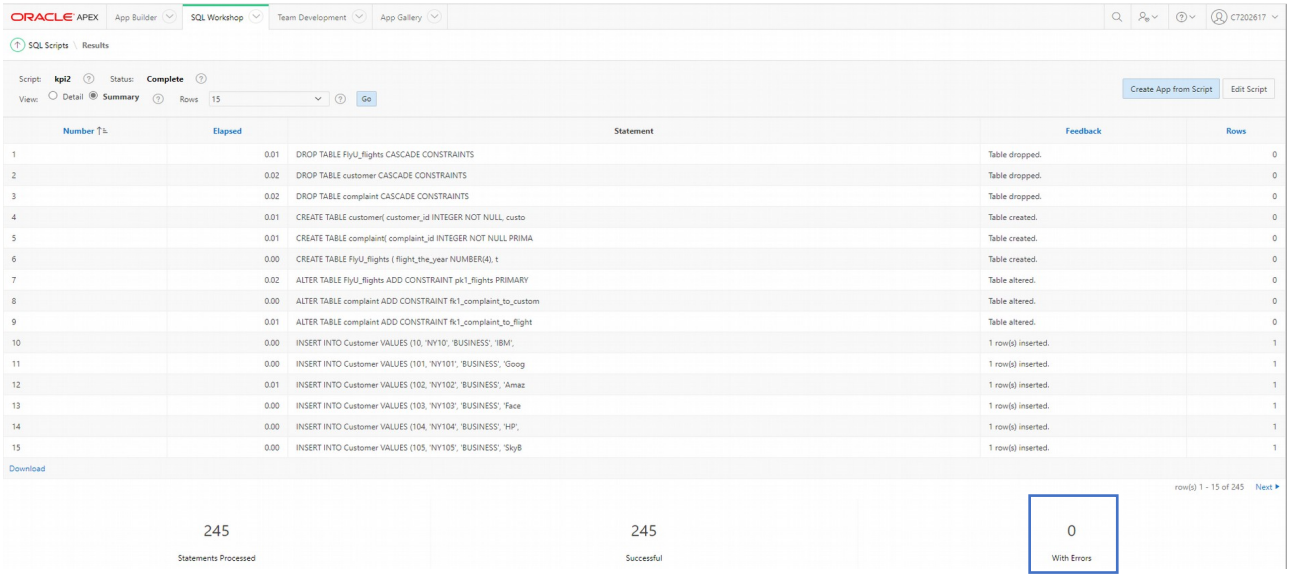


Task 2

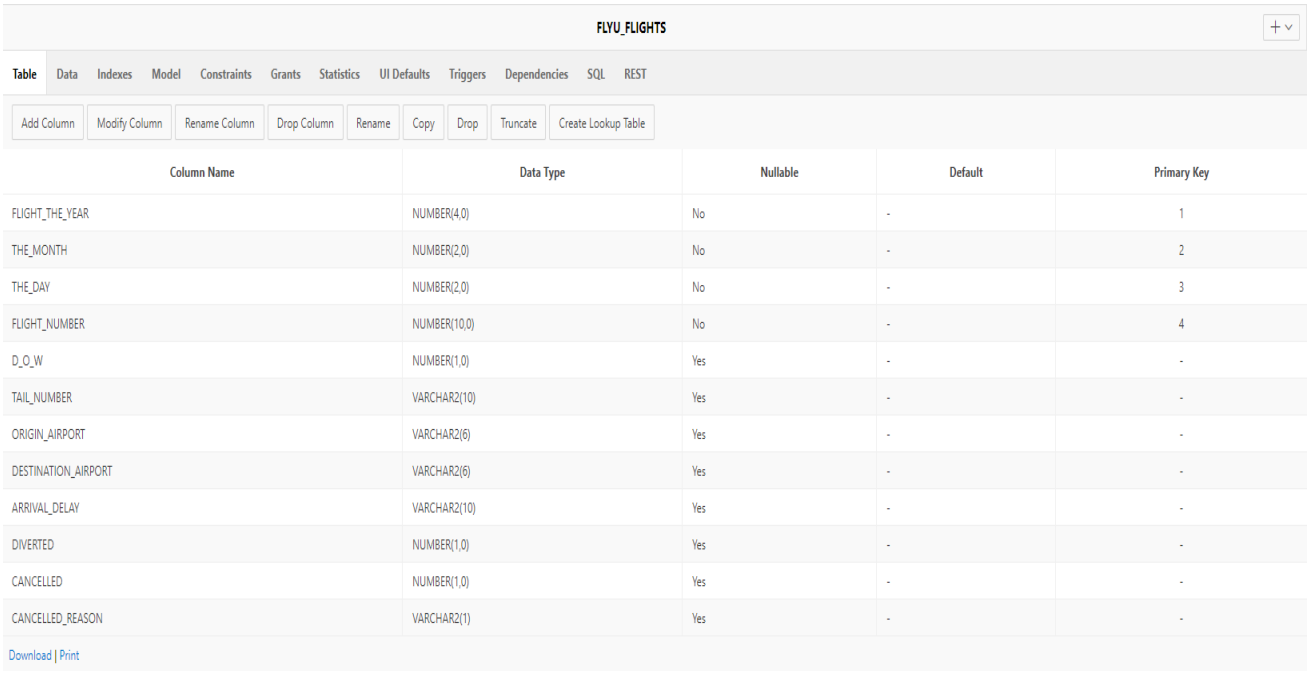
STAGING AREA SETUP:

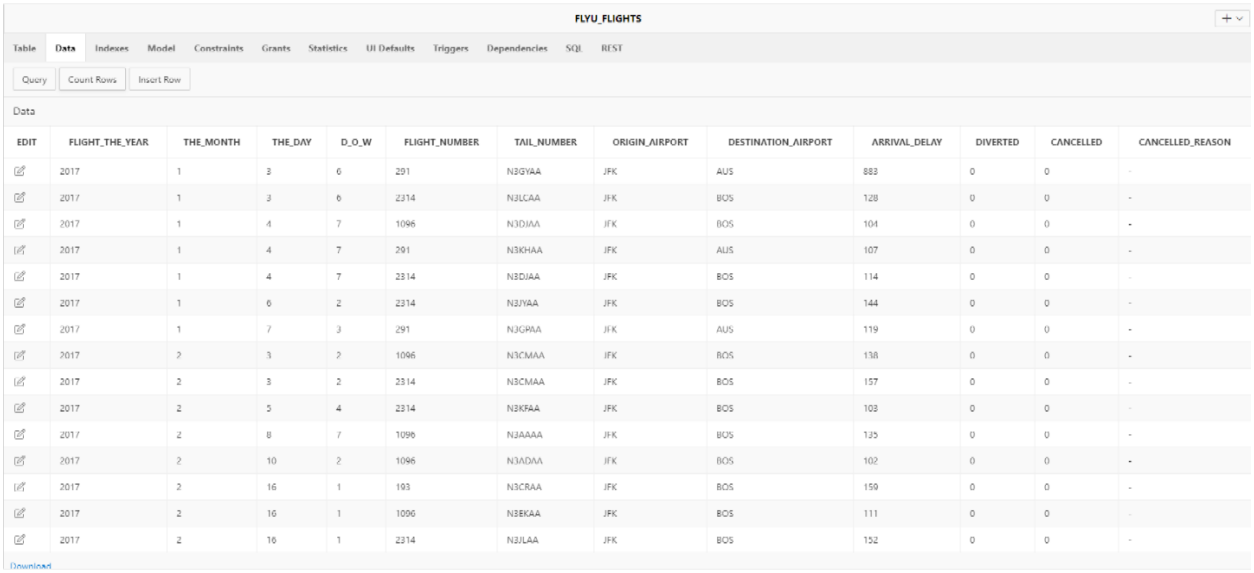
In this step, the multiple sources of data are extracted and run into the database. In this case, as we are dealing with complaints and customer service, we have only extracted flyU_flights data source. Hence we ended up with three tables: FlyU_flights, Complaints and Customers.

- Data Source



- FlyU_flights





FLYU_FLIGHTS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

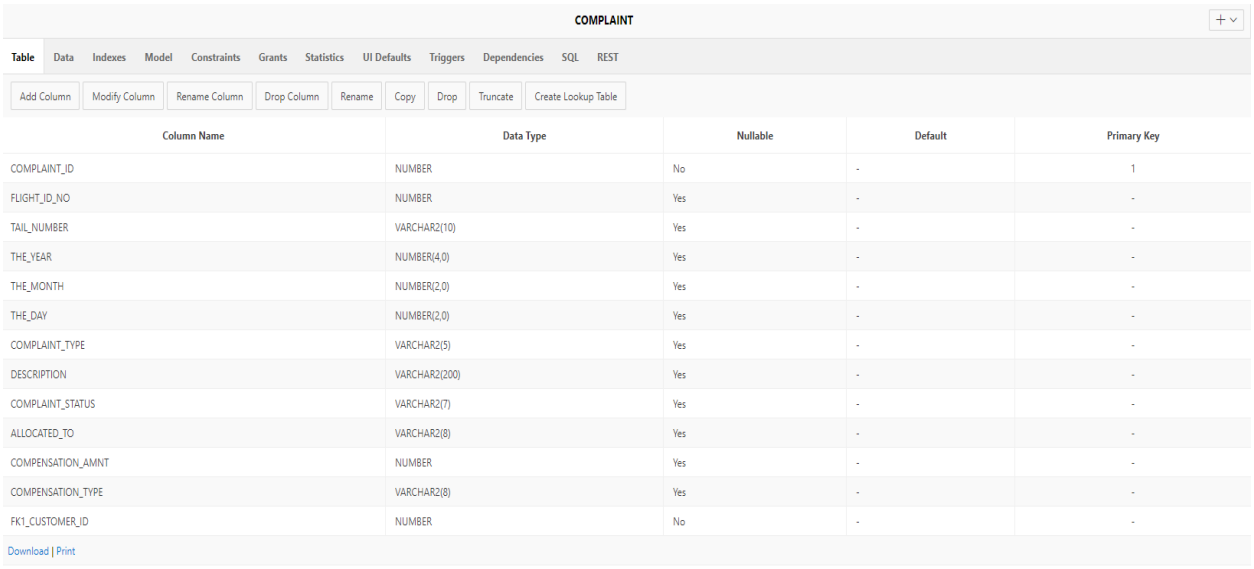
Query | Count Rows | Insert Row

Data

EDIT	FLIGHT_THE_YEAR	THE_MONTH	THE_DAY	D_O_W	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	ARRIVAL_DELAY	DIVERTED	CANCELLED	CANCELLED_REASON
	2017	1	3	6	291	N3GYAA	JFK	AUS	883	0	0	-
	2017	1	3	6	2314	N3LCAA	JFK	BOS	128	0	0	-
	2017	1	4	7	1096	N3DJAA	JFK	BOS	104	0	0	-
	2017	1	4	7	291	N3KHAA	JFK	AUS	107	0	0	-
	2017	1	4	7	2314	N3DJAA	JFK	BOS	114	0	0	-
	2017	1	6	2	2314	N3JYAA	JFK	BOS	144	0	0	-
	2017	1	7	3	291	N3GPAA	JFK	AUS	119	0	0	-
	2017	2	3	2	1096	N3CMAA	JFK	BOS	138	0	0	-
	2017	2	3	2	2314	N3CMAA	JFK	BOS	157	0	0	-
	2017	2	5	4	2314	N3KFAA	JFK	BOS	103	0	0	-
	2017	2	8	7	1096	N3AAAA	JFK	BOS	135	0	0	-
	2017	2	10	2	1096	N3ADAA	JFK	BOS	102	0	0	-
	2017	2	16	1	103	N3CRAA	JFK	BOS	159	0	0	-
	2017	2	16	1	1096	N3EKAA	JFK	BOS	111	0	0	-
	2017	2	16	1	2314	N3JLAA	JFK	BOS	152	0	0	-

Download

- Complaint



COMPLAINT

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
COMPLAINT_ID	NUMBER	No	-	1
FLIGHT_ID_NO	NUMBER	Yes	-	-
TAIL_NUMBER	VARCHAR2(10)	Yes	-	-
THE_YEAR	NUMBER(4,0)	Yes	-	-
THE_MONTH	NUMBER(2,0)	Yes	-	-
THE_DAY	NUMBER(2,0)	Yes	-	-
COMPLAINT_TYPE	VARCHAR2(5)	Yes	-	-
DESCRIPTION	VARCHAR2(200)	Yes	-	-
COMPLAINT_STATUS	VARCHAR2(7)	Yes	-	-
ALLOCATED_TO	VARCHAR2(8)	Yes	-	-
COMPENSATION_AMNT	NUMBER	Yes	-	-
COMPENSATION_TYPE	VARCHAR2(8)	Yes	-	-
FK1_CUSTOMER_ID	NUMBER	No	-	-

Download | Print

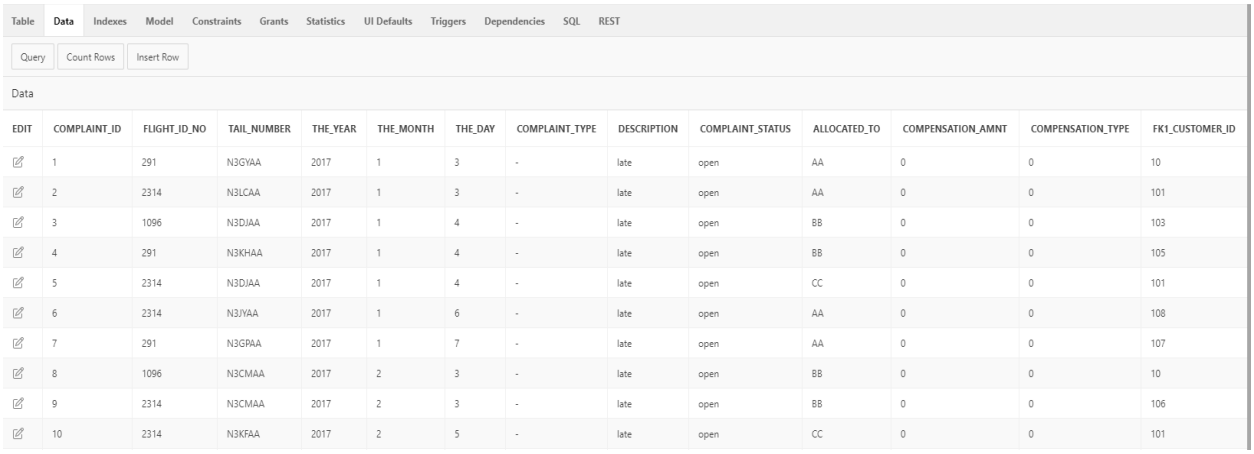


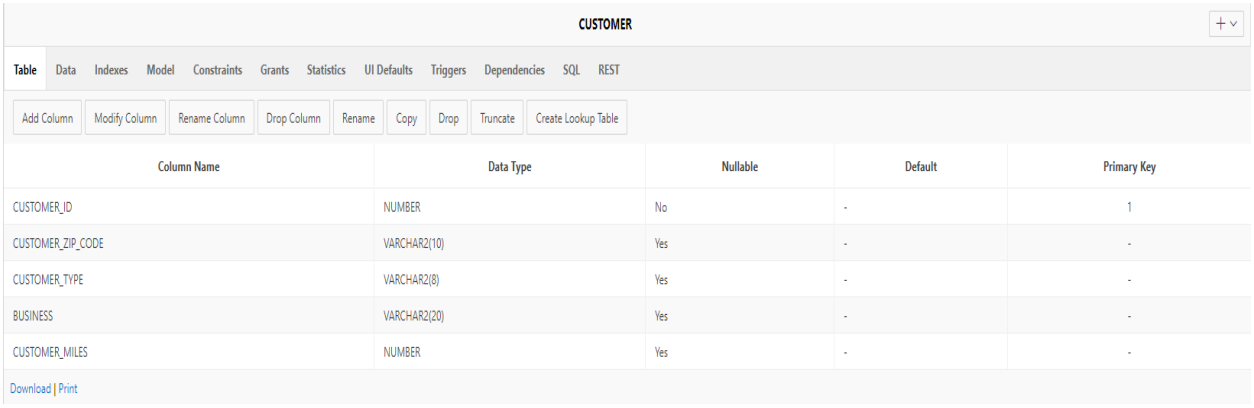
Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Query | Count Rows | Insert Row

Data

EDIT	COMPLAINT_ID	FLIGHT_ID_NO	TAIL_NUMBER	THE_YEAR	THE_MONTH	THE_DAY	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	ALLOCATED_TO	COMPENSATION_AMNT	COMPENSATION_TYPE	FK1_CUSTOMER_ID
	1	291	N3GYAA	2017	1	3	-	late	open	AA	0	0	10
	2	2314	N3LCAA	2017	1	3	-	late	open	AA	0	0	101
	3	1096	N3DJAA	2017	1	4	-	late	open	BB	0	0	103
	4	291	N3KHAA	2017	1	4	-	late	open	BB	0	0	105
	5	2314	N3DJAA	2017	1	4	-	late	open	CC	0	0	101
	6	2314	N3JYAA	2017	1	6	-	late	open	AA	0	0	108
	7	291	N3GPAA	2017	1	7	-	late	open	AA	0	0	107
	8	1096	N3CMAA	2017	2	3	-	late	open	BB	0	0	10
	9	2314	N3CMAA	2017	2	3	-	late	open	BB	0	0	106
	10	2314	N3KFAA	2017	2	5	-	late	open	CC	0	0	101

- Customer



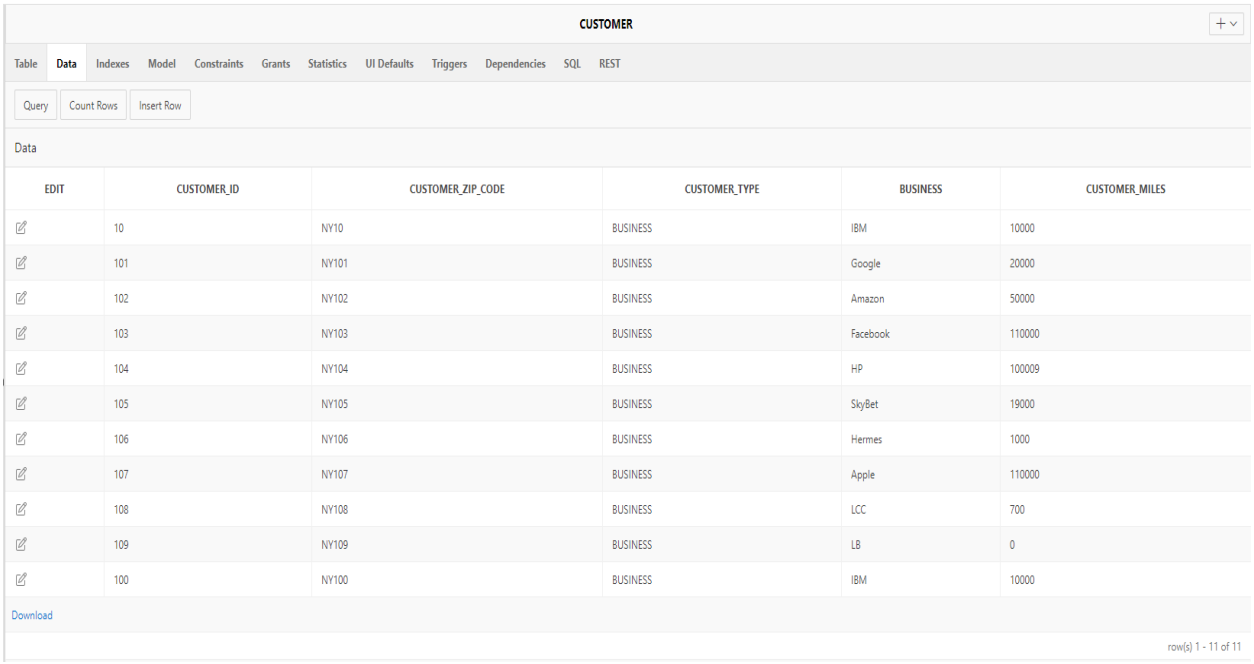
CUSTOMER

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMER_ID	NUMBER	No	-	1
CUSTOMER_ZIP_CODE	VARCHAR2(10)	Yes	-	-
CUSTOMER_TYPE	VARCHAR2(8)	Yes	-	-
BUSINESS	VARCHAR2(20)	Yes	-	-
CUSTOMER_MILES	NUMBER	Yes	-	-

Download | Print



CUSTOMER

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Query | Count Rows | Insert Row

Data

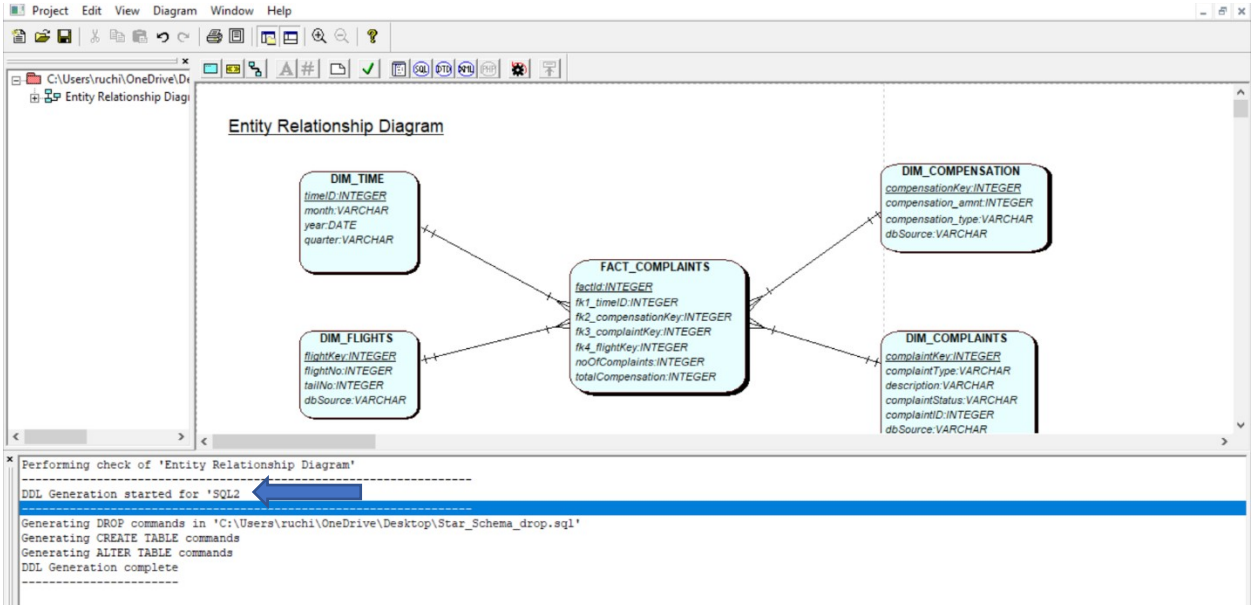
EDIT	CUSTOMER_ID	CUSTOMER_ZIP_CODE	CUSTOMER_TYPE	BUSINESS	CUSTOMER_MILES
	10	NY10	BUSINESS	IBM	10000
	101	NY101	BUSINESS	Google	20000
	102	NY102	BUSINESS	Amazon	50000
	103	NY103	BUSINESS	Facebook	110000
	104	NY104	BUSINESS	HP	100009
	105	NY105	BUSINESS	SkyBet	19000
	106	NY106	BUSINESS	Hermes	1000
	107	NY107	BUSINESS	Apple	110000
	108	NY108	BUSINESS	LCC	700
	109	NY109	BUSINESS	LB	0
	100	NY100	BUSINESS	IBM	10000

Download

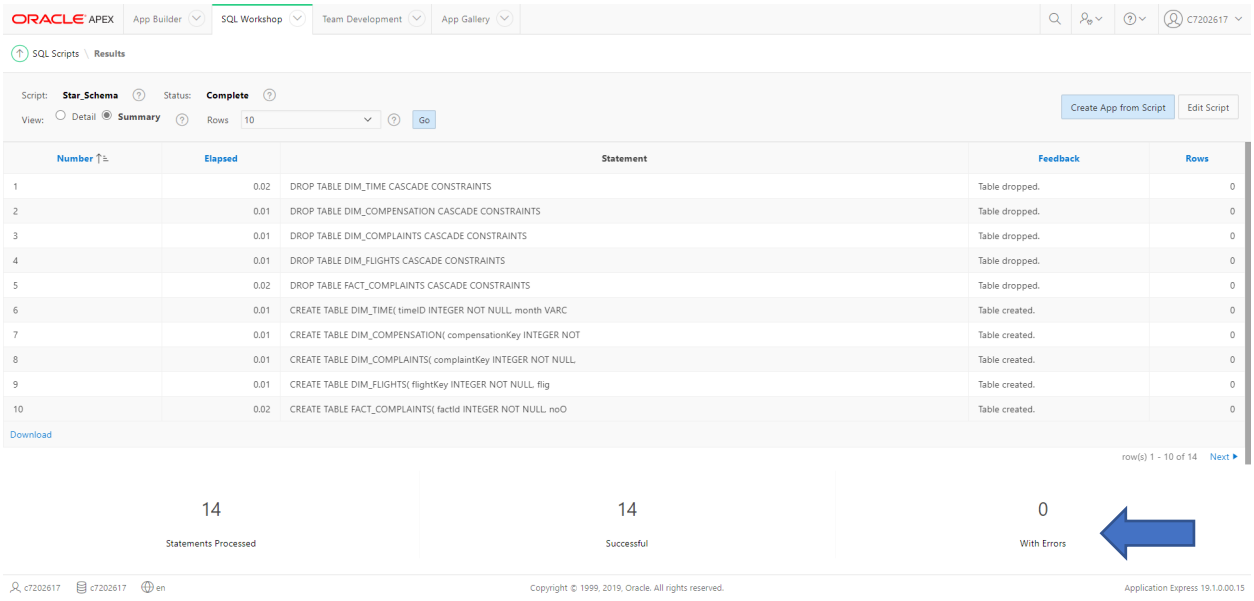
row(s) 1 - 11 of 11

STAR SCHEMA SETUP:

We have created a star schema based on the reports created.



The script of the star schema is forward engineered using Qsee tool and uploaded into the database.

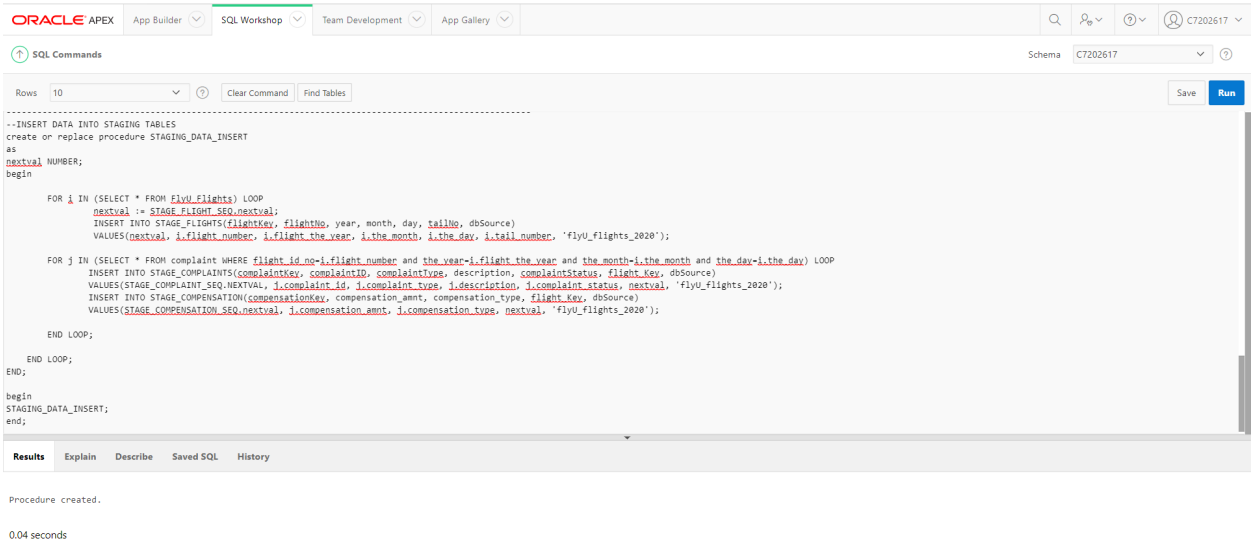


Task 3

This part of the project deals with building the data warehouse using the **ETL process**. It is a type of data integration completed in three steps: extract, transform, load.

EXTRACTION:

In this part of the ETL process, we have extracted data from the data sources and transferred into the staging area.

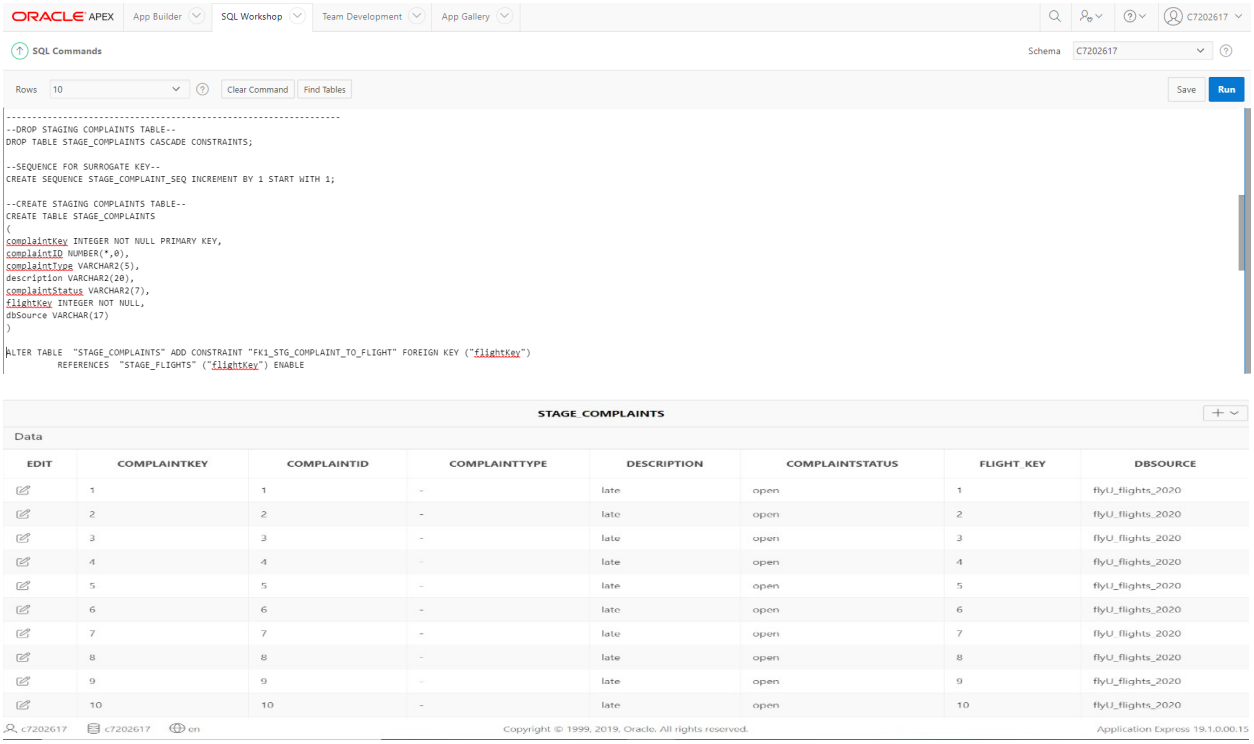


```
--INSERT DATA INTO STAGING TABLES
create or replace procedure STAGING_DATA_INSERT
as
nextval NUMBER;
begin

       FOR i IN (SELECT * FROM flyU_Flights) LOOP
                nextval := STAGE_FLIGHT_SEQ.nextval;
                INSERT INTO STAGE_FLIGHTS(flightKey, flightNo, year, month, day, tailNo, dbSource)
                VALUES(nextval, i.flight_number, i.flight_the_year, i.the_month, i.the_day, i.tail_number, 'flyU_flights_2020');

       FOR j IN (SELECT * FROM complaint WHERE flight_id_no=i.flight_number and the_year=i.flight_the_year and the_month=i.the_month and the_day=i.the_day) LOOP
              INSERT INTO STAGE_COMPLAINTS(complaintKey, complaintID, complaintType, description, complaintStatus, flight_key, dbSource)
              VALUES(STAGE_COMPLAINT_SEQ.NEXTVAL, j.complaint_id, j.complaint_type, j.description, j.complaint_status, nextval, 'flyU_flights_2020');
              INSERT INTO STAGE_COMPENSATION(compensationKey, compensation_amnt, compensation_type, flight_Key, dbSource)
              VALUES(STAGE_COMPENSATION_SEQ.nextval, j.compensation_amnt, j.compensation_type, nextval, 'flyU_flights_2020');

       END LOOP;

   END LOOP;
END;

begin
STAGING_DATA_INSERT;
end;
```

```
Procedure created.

0.04 seconds
```

There are three staging tables:

- **Stage Complaints**



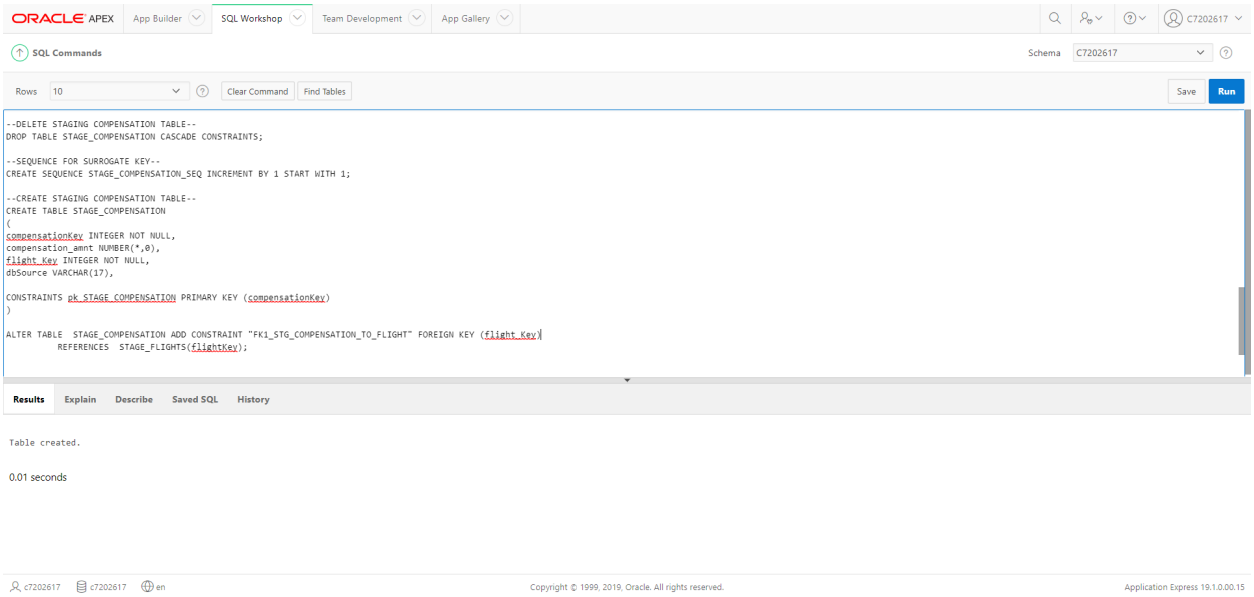
```
--DROP STAGING COMPLAINTS TABLE--
DROP TABLE STAGE_COMPLAINTS CASCADE CONSTRAINTS;

--SEQUENCE FOR SURROGATE KEY--
CREATE SEQUENCE STAGE_COMPLAINT_SEQ INCREMENT BY 1 START WITH 1;

--CREATE STAGING COMPLAINTS TABLE--
CREATE TABLE STAGE_COMPLAINTS
(
complaintKey INTEGER NOT NULL PRIMARY KEY,
complaintID NUMBER(*,0),
complaintType VARCHAR2(5),
description VARCHAR2(20),
complaintStatus VARCHAR2(7),
flightKey INTEGER NOT NULL,
dbSource VARCHAR(17)
)

ALTER TABLE  "STAGE_COMPLAINTS" ADD CONSTRAINT "FK1_STG_COMPLAINT_TO_FLIGHT" FOREIGN KEY ("flightKey")
          REFERENCES  "STAGE_FLIGHTS" ("flightKey") ENABLE
```

STAGE_COMPLAINTS

Data

EDIT	COMPLAINTKEY	COMPLAINTID	COMPLAINTTYPE	DESCRIPTION	COMPLAINTSTATUS	FLIGHT_KEY	DBSOURCE
	1	1	-	late	open	1	flyU_flights_2020
	2	2	-	late	open	2	flyU_flights_2020
	3	3	-	late	open	3	flyU_flights_2020
	4	4	-	late	open	4	flyU_flights_2020
	5	5	-	late	open	5	flyU_flights_2020
	6	6	-	late	open	6	flyU_flights_2020
	7	7	-	late	open	7	flyU_flights_2020
	8	8	-	late	open	8	flyU_flights_2020
	9	9	-	late	open	9	flyU_flights_2020
	10	10	-	late	open	10	flyU_flights_2020

- **Stage Compensation**

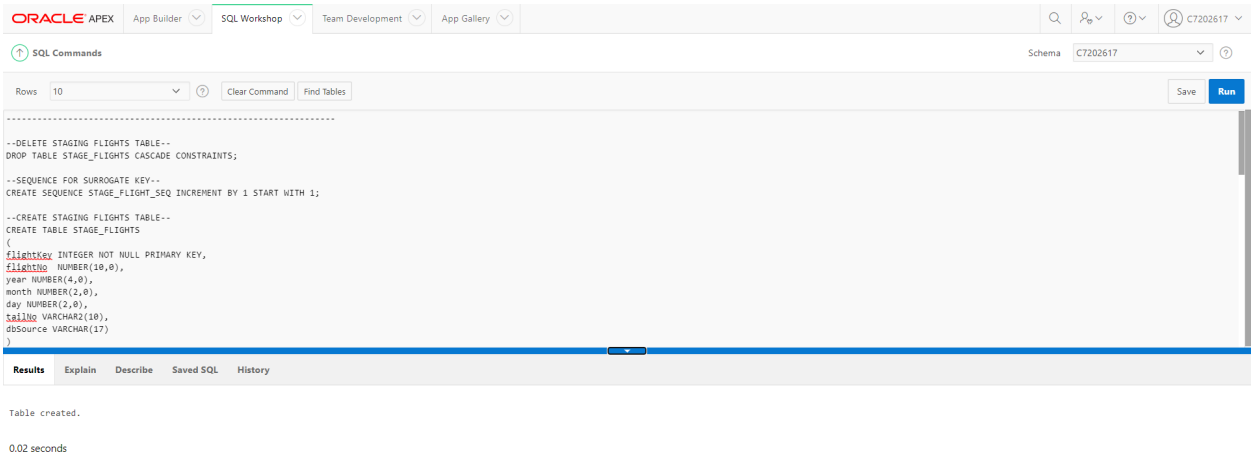


STAGE_COMPENSATION

EDIT	COMPENSATIONKEY	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	1	0	0	1	flyU_flights_2020
	2	0	0	2	flyU_flights_2020
	3	0	0	3	flyU_flights_2020
	4	0	0	4	flyU_flights_2020
	5	0	0	5	flyU_flights_2020
	6	0	0	6	flyU_flights_2020
	7	0	0	7	flyU_flights_2020
	8	0	0	8	flyU_flights_2020
	9	0	0	9	flyU_flights_2020
	10	0	0	10	flyU_flights_2020
	11	0	0	11	flyU_flights_2020

c7202617 c7202617 en
Copyright © 1999, 2019, Oracle. All rights reserved.
Application Express 19.1.0.00.15

- **Stage Flights**



STAGE_FLIGHTS

EDIT	FLIGHTKEY	FLIGHTNO	YEAR	MONTH	DAY	TAILNO	DBSOURCE
	1	291	2017	1	3	N3GYAA	flyU_flights_2020
	2	2314	2017	1	3	N3LCAA	flyU_flights_2020
	3	1096	2017	1	4	N3DJAA	flyU_flights_2020
	4	291	2017	1	4	N3KHAA	flyU_flights_2020
	5	2314	2017	1	4	N3DJAA	flyU_flights_2020
	6	2314	2017	1	6	N3JYAA	flyU_flights_2020
	7	291	2017	1	7	N3GPAA	flyU_flights_2020
	8	1096	2017	2	3	N3CMAA	flyU_flights_2020
	9	2314	2017	2	3	N3CMAA	flyU_flights_2020
	10	2314	2017	2	5	N3KFAA	flyU_flights_2020
	11	1096	2017	2	8	N3AAAA	flyU_flights_2020

c7202617 c7202617 en

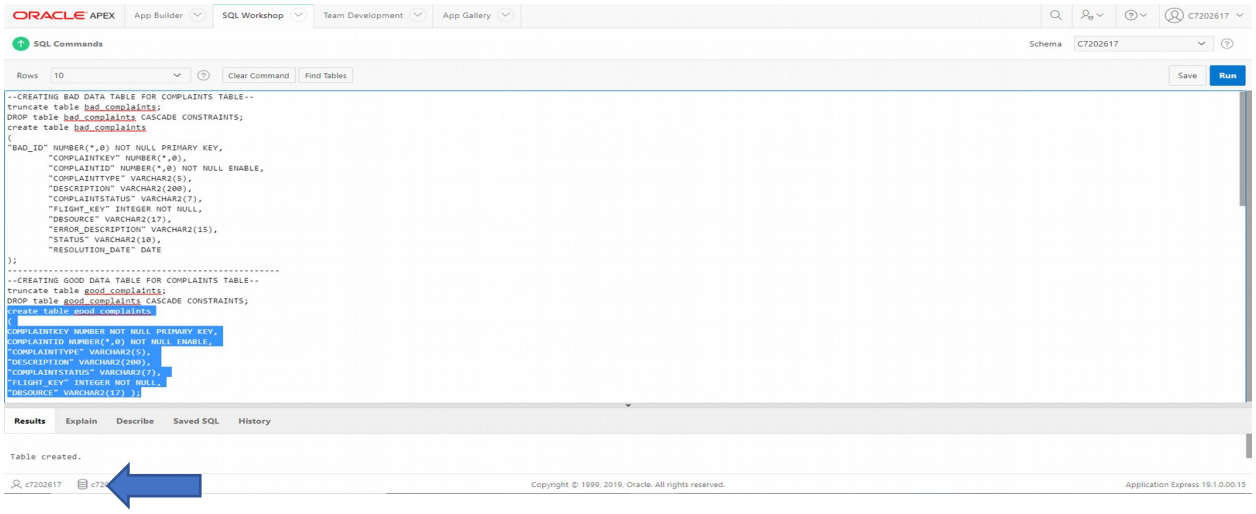
Application Express 19.1.0.00.15

DATA PURIFICATION:

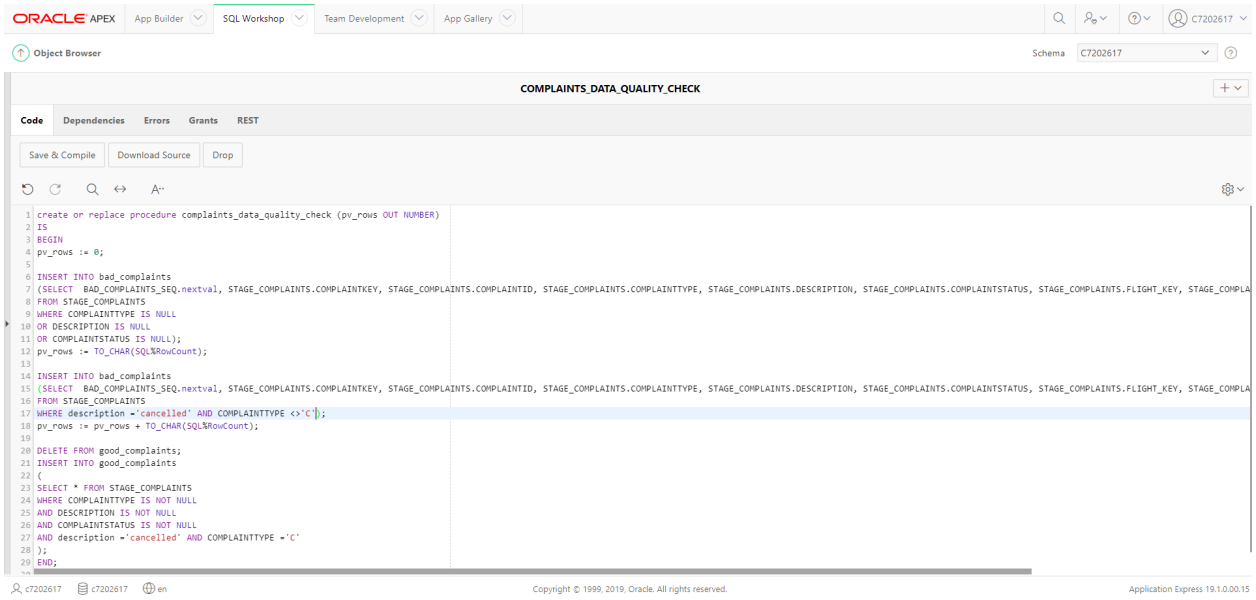
In this part of the ETL process, we identify all the bad and good data's determined in the data dictionary, create good and bad tables and populate it, reclean bad data and transfer the bad data into the good tables.

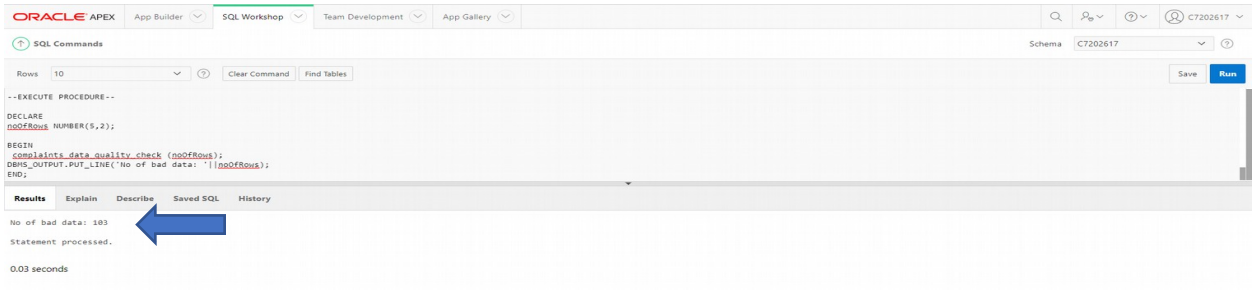
COMPLAINT

- Table Creation

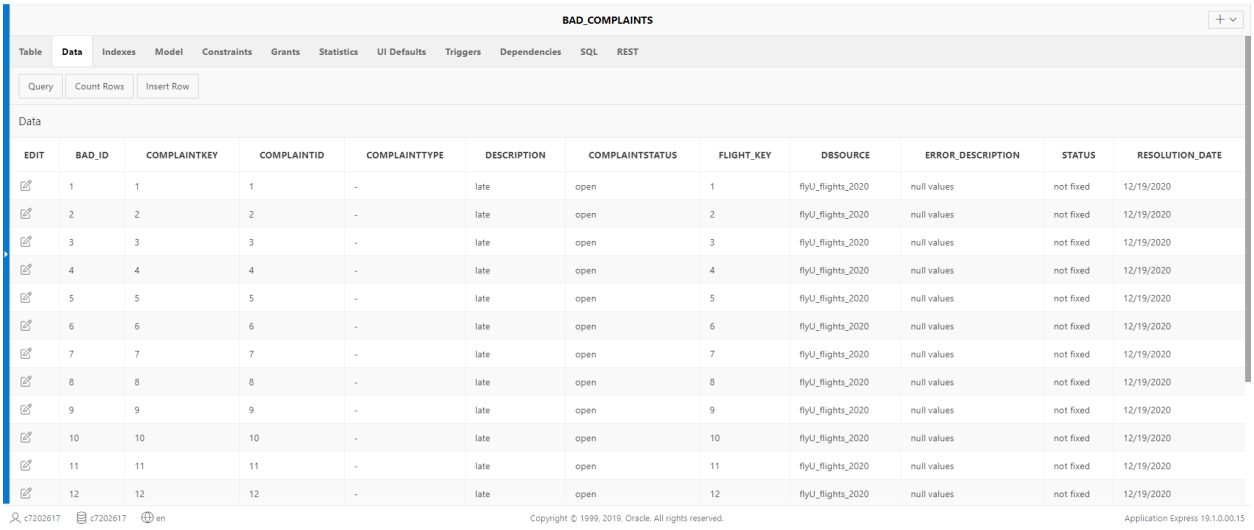


- Bad/Good Data Identification

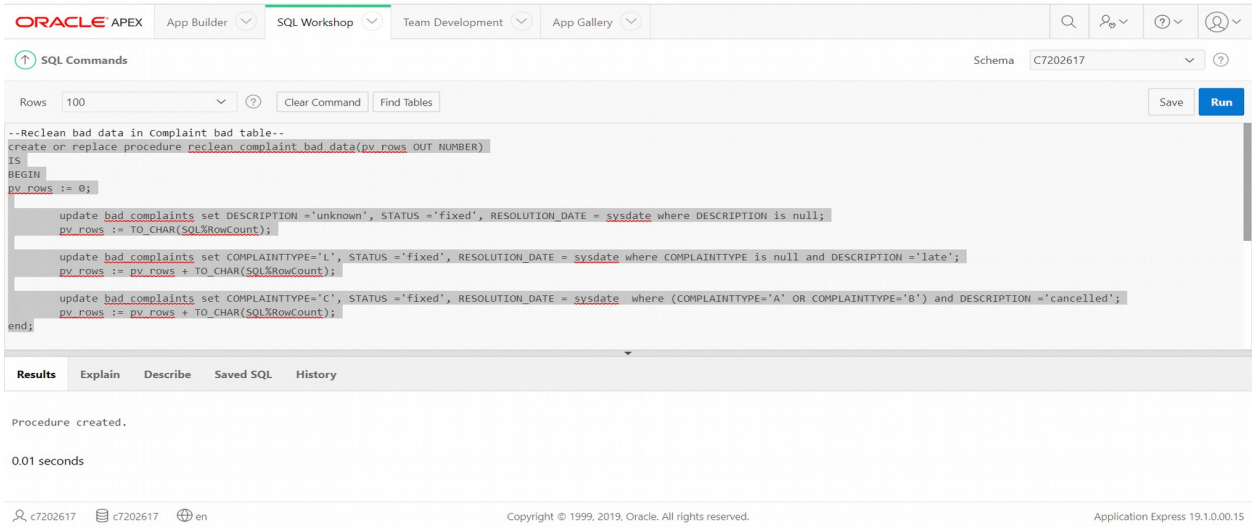




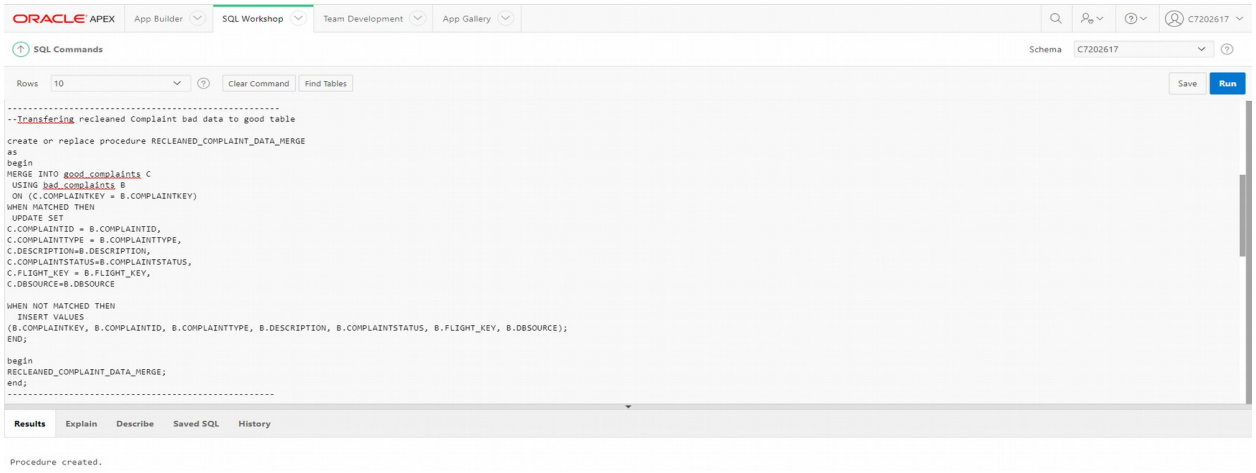
- BAD TABLE



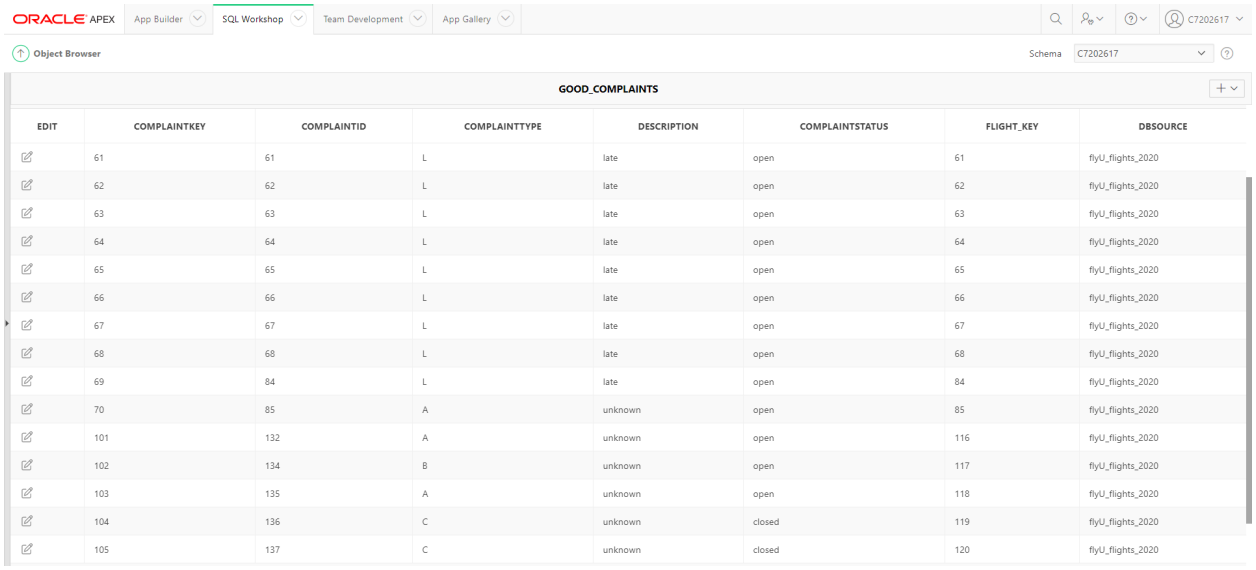
- Reclean Bad Data



- Transfer Recleaned Data



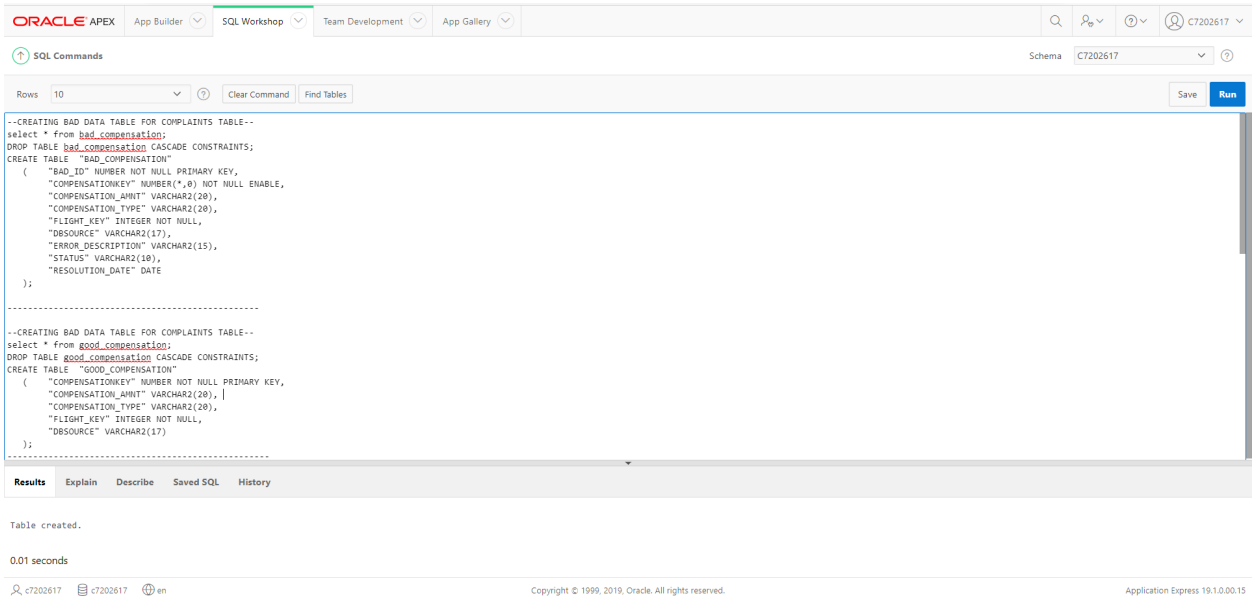
- Good Table



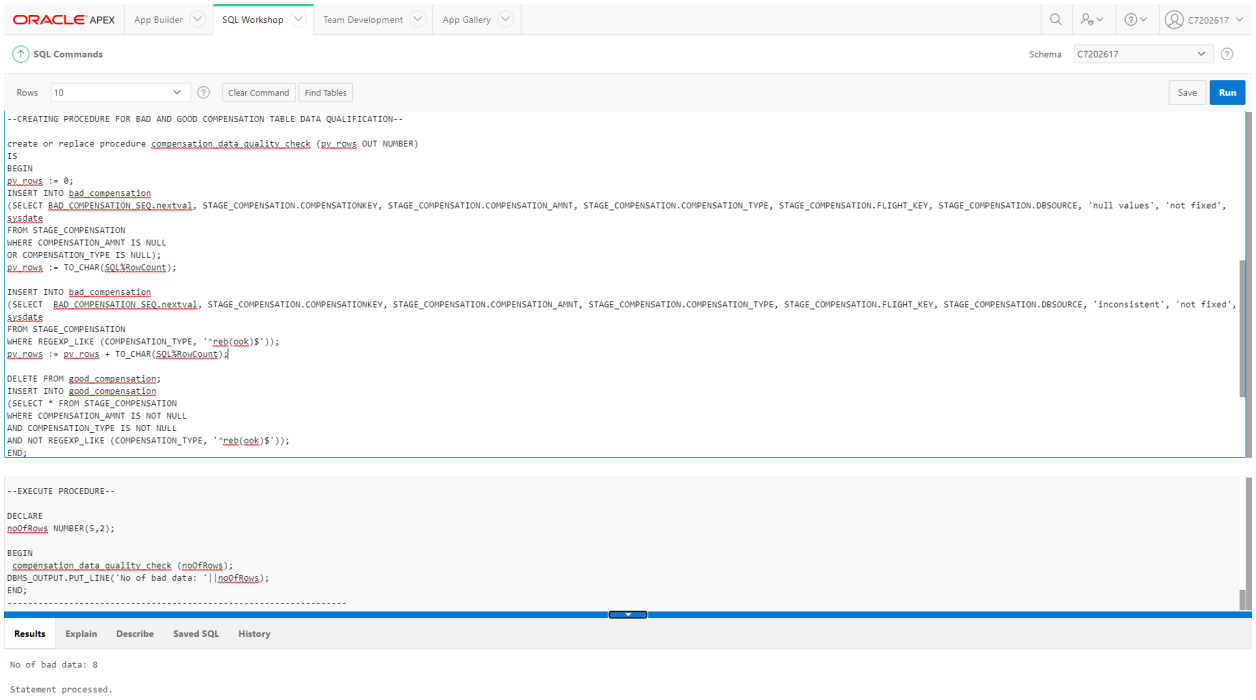
EDIT	COMPLAINTKEY	COMPLAINTID	COMPLAINTTYPE	DESCRIPTION	COMPLAINTSTATUS	FLIGHT_KEY	DBSOURCE
	61	61	L	late	open	61	flyU_flights_2020
	62	62	L	late	open	62	flyU_flights_2020
	63	63	L	late	open	63	flyU_flights_2020
	64	64	L	late	open	64	flyU_flights_2020
	65	65	L	late	open	65	flyU_flights_2020
	66	66	L	late	open	66	flyU_flights_2020
	67	67	L	late	open	67	flyU_flights_2020
	68	68	L	late	open	68	flyU_flights_2020
	69	84	L	late	open	84	flyU_flights_2020
	70	85	A	unknown	open	85	flyU_flights_2020
	101	132	A	unknown	open	116	flyU_flights_2020
	102	134	B	unknown	open	117	flyU_flights_2020
	103	135	A	unknown	open	118	flyU_flights_2020
	104	136	C	unknown	closed	119	flyU_flights_2020
	105	137	C	unknown	closed	120	flyU_flights_2020

COMPENSATION

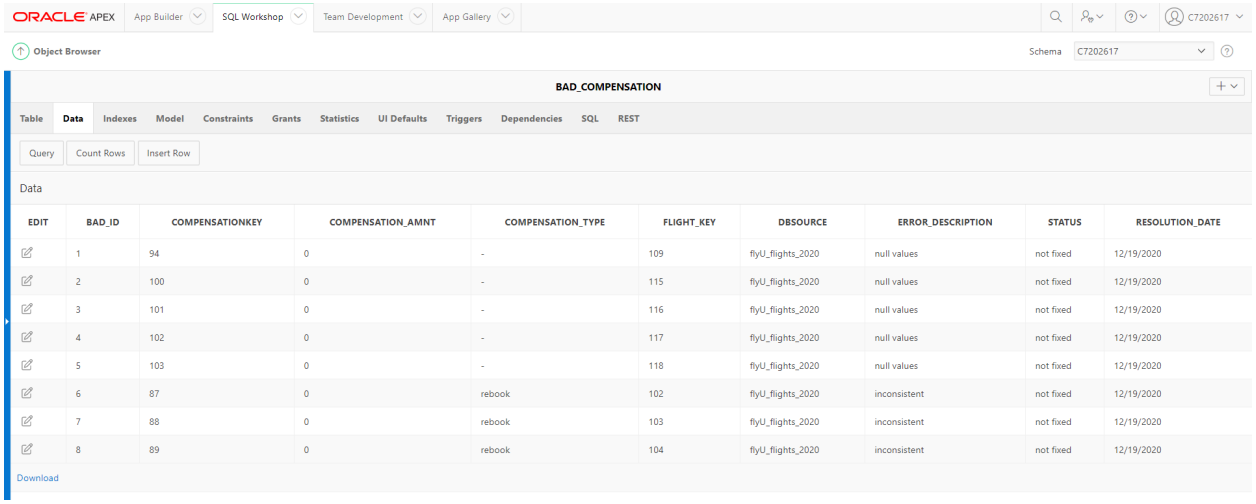
- Table Creation



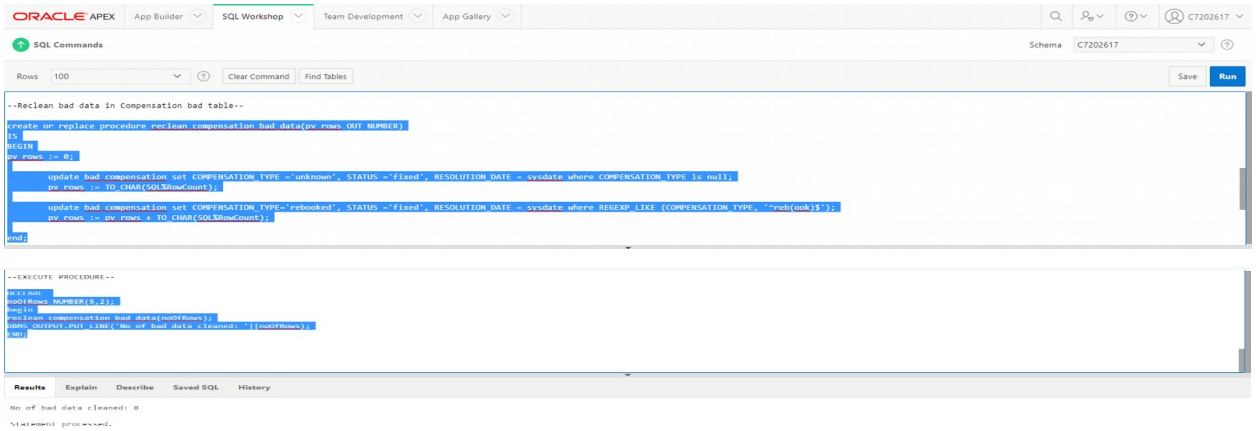
- Bad/Good Data Identification



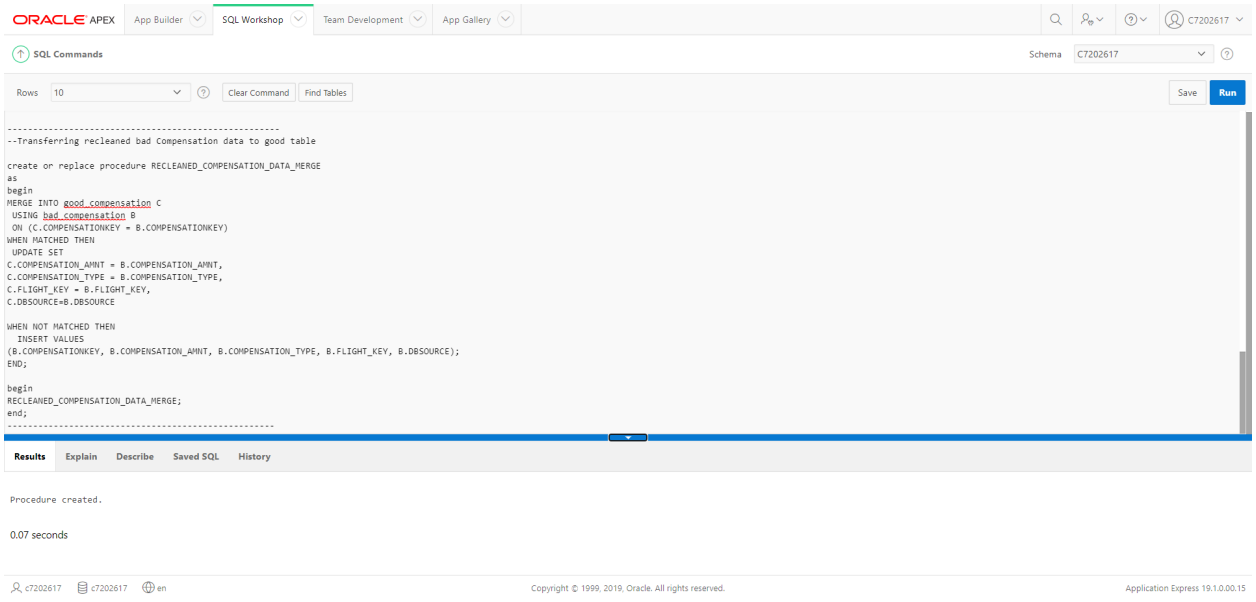
- BAD TABLE



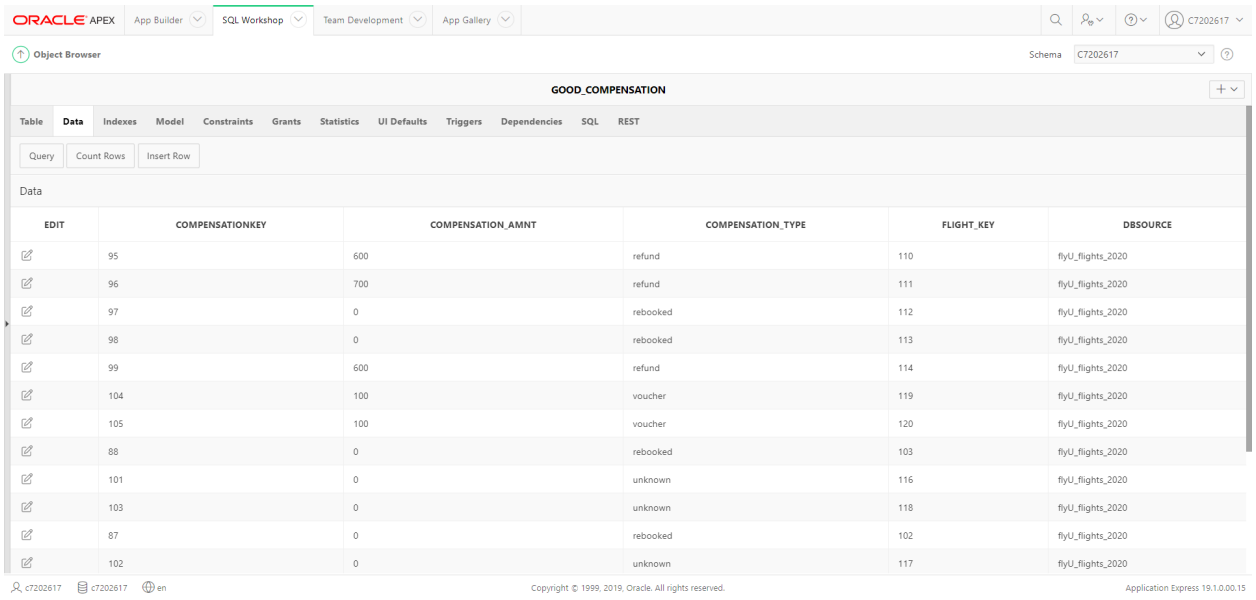
- Reclean Bad Data



- Transfer Recleaned Data

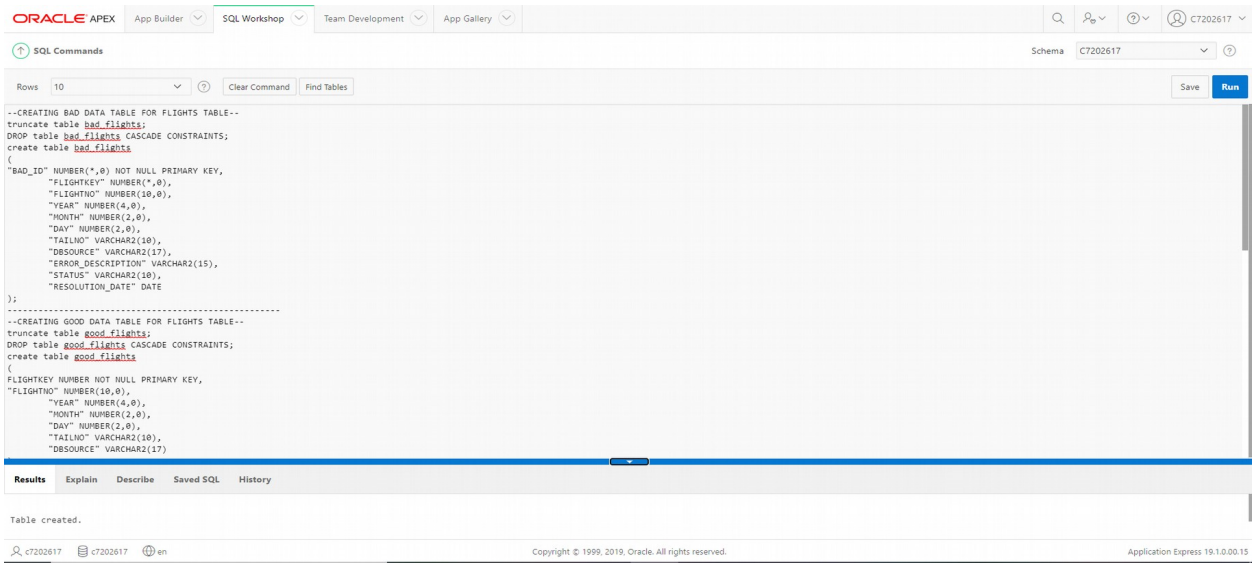


- Good Table

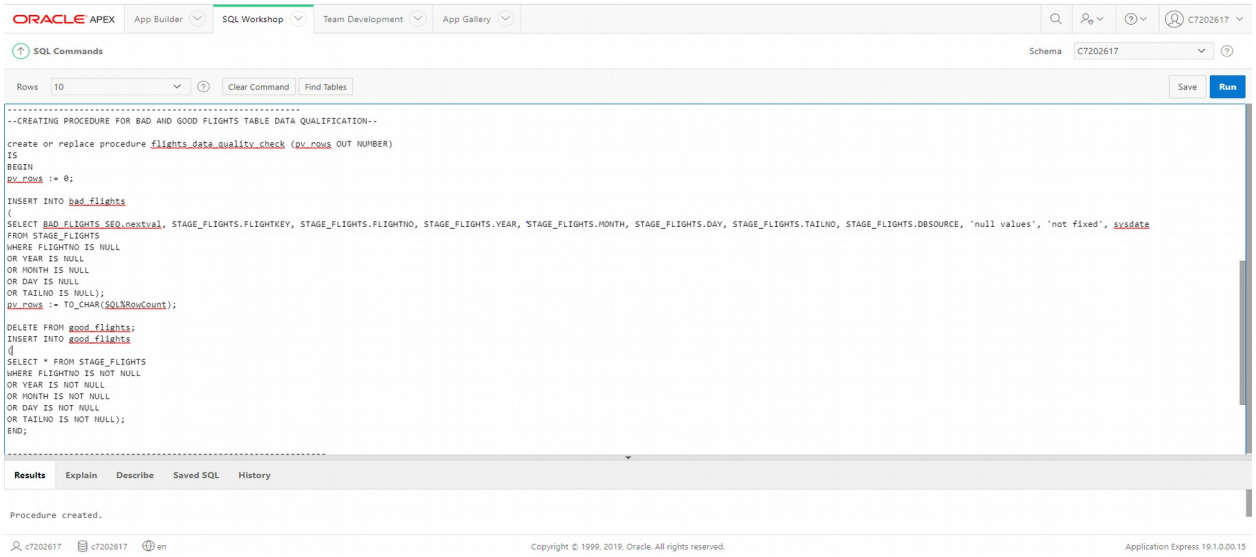


FLIGHTS

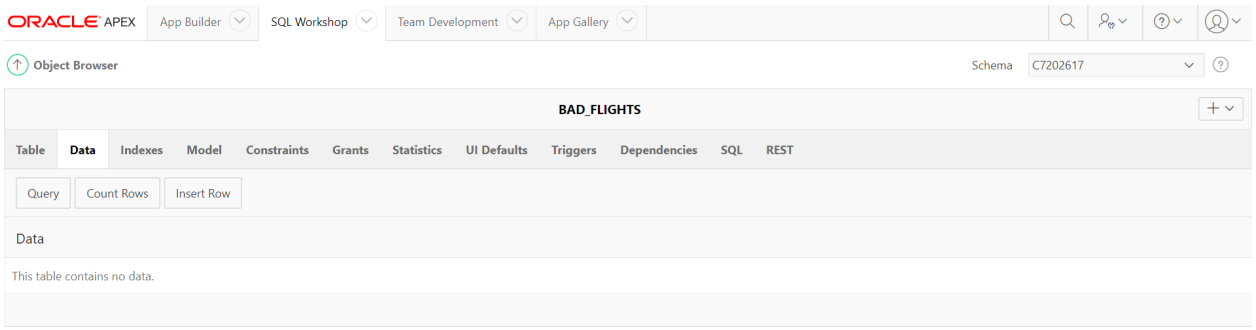
- Table Creation



- Bad/Good Data Identification



- BAD TABLE



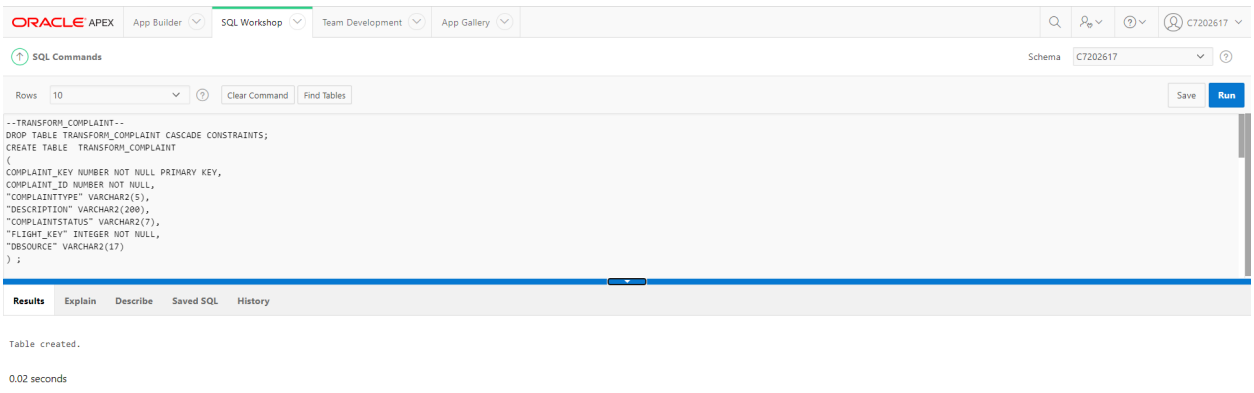
- Reclean Bad Data

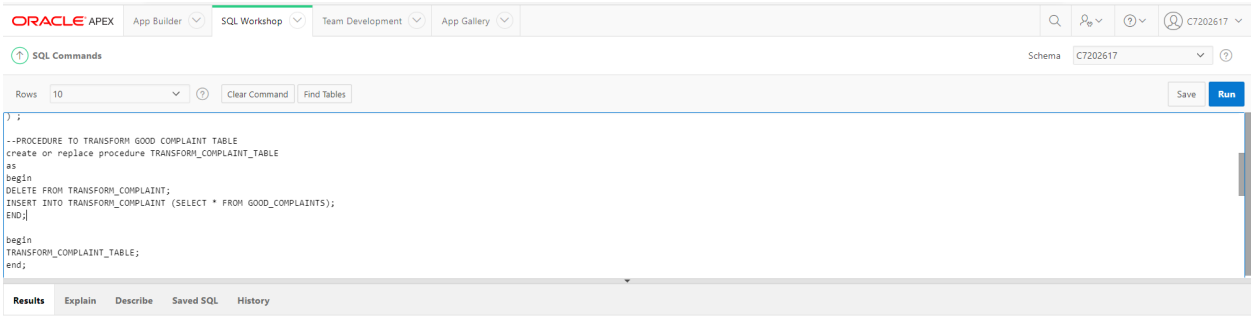
  There is no bad data to be cleaned for flights table.

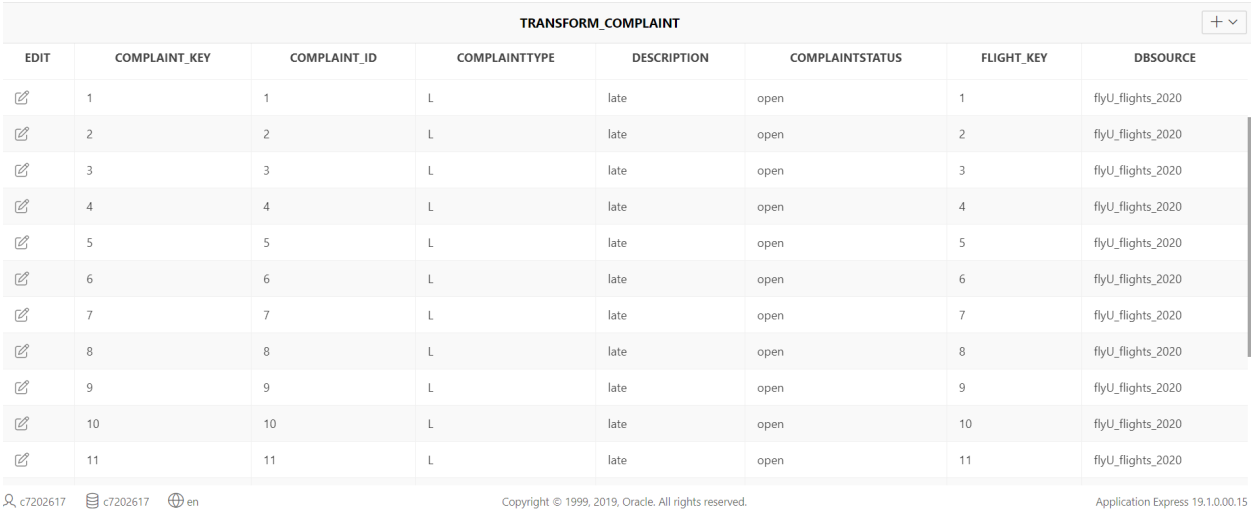
TRANSFORMATION:

In this part of the ETL process, we transform all the cleaned data from the clean tables into the their respective transformation table.

- **Transform Complaints**



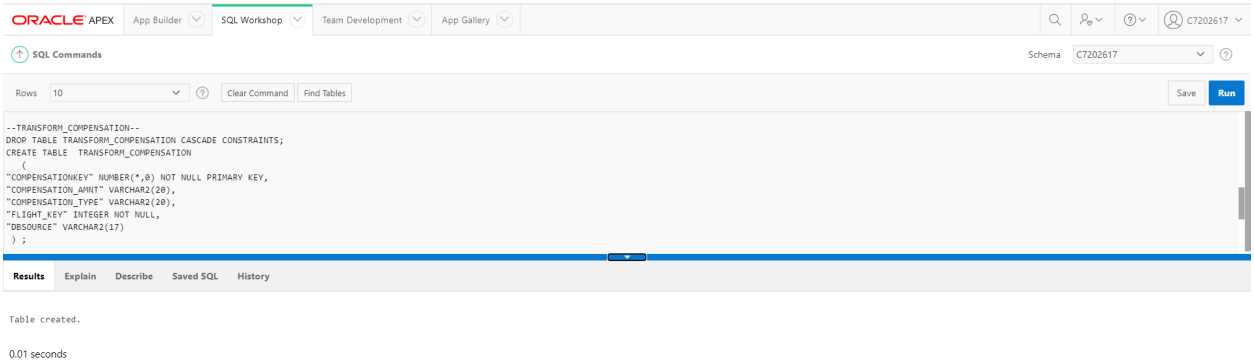




TRANSFORM_COMPLAINT

EDIT	COMPLAINT_KEY	COMPLAINT_ID	COMPLAINTTYPE	DESCRIPTION	COMPLAINTSTATUS	FLIGHT_KEY	DBSOURCE
	1	1	L	late	open	1	flyU_flights_2020
	2	2	L	late	open	2	flyU_flights_2020
	3	3	L	late	open	3	flyU_flights_2020
	4	4	L	late	open	4	flyU_flights_2020
	5	5	L	late	open	5	flyU_flights_2020
	6	6	L	late	open	6	flyU_flights_2020
	7	7	L	late	open	7	flyU_flights_2020
	8	8	L	late	open	8	flyU_flights_2020
	9	9	L	late	open	9	flyU_flights_2020
	10	10	L	late	open	10	flyU_flights_2020
	11	11	L	late	open	11	flyU_flights_2020

- **Transform Compensation**



```
--PROCEDURE TO TRANSFORM GOOD COMPENSATION TABLE
create or replace procedure TRANSFORM_COMPENSATION_TABLE
as
begin
DELETE FROM TRANSFORM_COMPENSATION;
INSERT INTO TRANSFORM_COMPENSATION (SELECT * FROM GOOD_COMPENSATION);
END;

begin
TRANSFORM_COMPENSATION_TABLE;
end;
```

Results | Explain | Describe | Saved SQL | History

Procedure created.

0.01 seconds

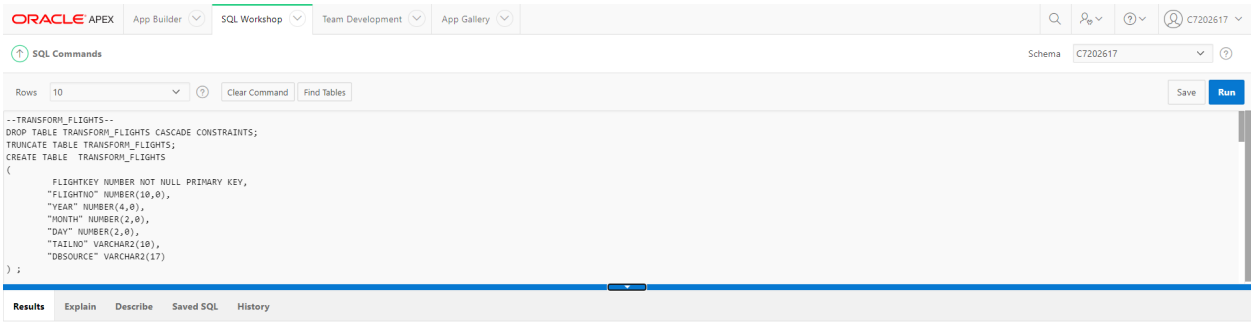
TRANSFORM_COMPENSATION

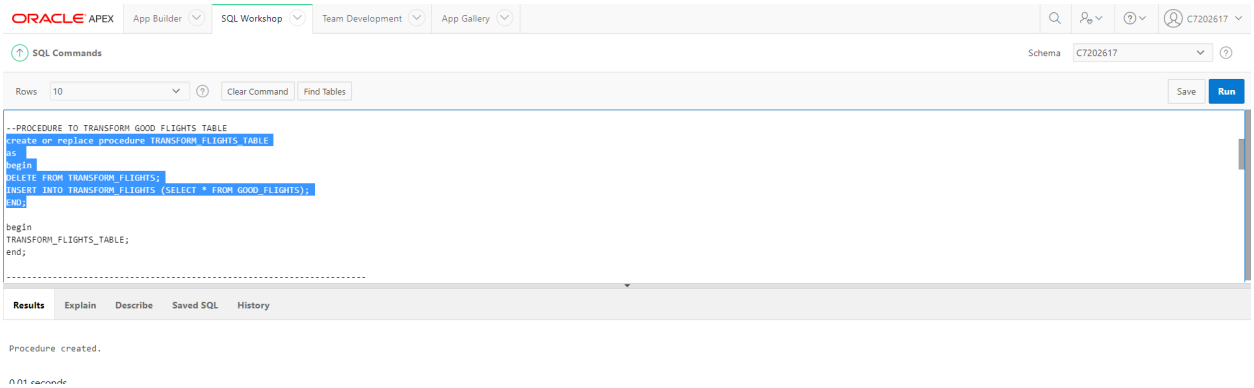
EDIT	COMPENSATIONKEY	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	1	0	0	1	flyU_flights_2020
	2	0	0	2	flyU_flights_2020
	3	0	0	3	flyU_flights_2020
	4	0	0	4	flyU_flights_2020
	5	0	0	5	flyU_flights_2020
	6	0	0	6	flyU_flights_2020
	7	0	0	7	flyU_flights_2020
	8	0	0	8	flyU_flights_2020
	9	0	0	9	flyU_flights_2020
	10	0	0	10	flyU_flights_2020
	11	0	0	11	flyU_flights_2020

c7202617 c7202617 en

Application Express 19.1.0.00.15

- **Transform Flights**



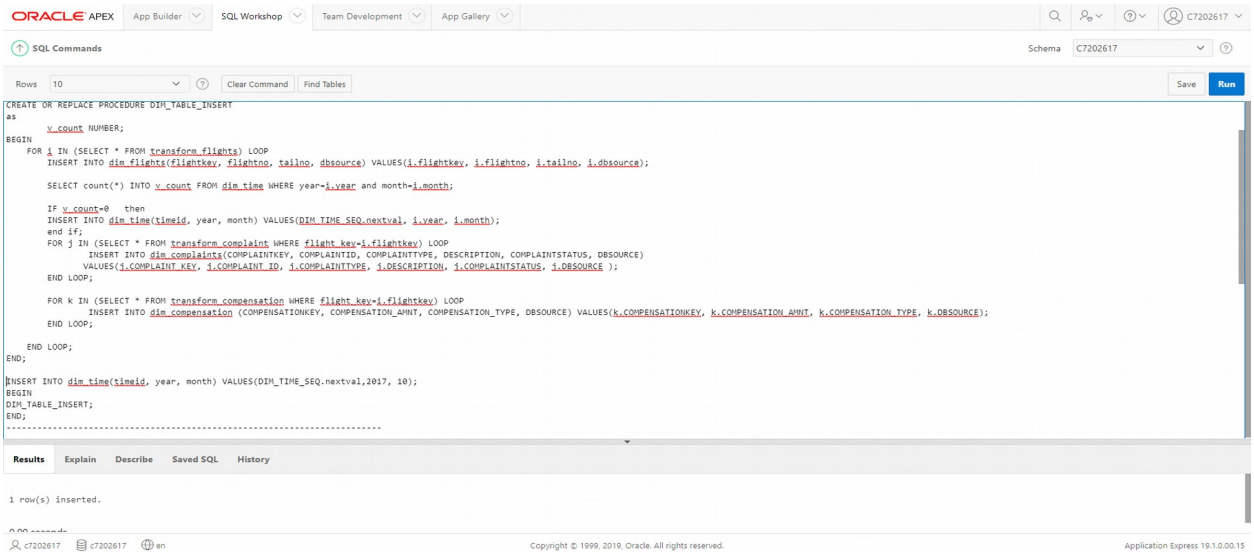


TRANSFORM_FLIGHTS

EDIT	FLIGHTKEY	FLIGHTNO	YEAR	MONTH	DAY	TAILNO	DBSOURCE
	1	291	2017	1	3	N3GYAA	flyU_flights_2020
	2	2314	2017	1	3	N3LCAA	flyU_flights_2020
	3	1096	2017	1	4	N3DJAA	flyU_flights_2020
	4	291	2017	1	4	N3KHAA	flyU_flights_2020
	5	2314	2017	1	4	N3DJAA	flyU_flights_2020
	6	2314	2017	1	6	N3JYAA	flyU_flights_2020
	7	291	2017	1	7	N3GPAA	flyU_flights_2020
	8	1096	2017	2	3	N3CMAA	flyU_flights_2020
	9	2314	2017	2	3	N3CMAA	flyU_flights_2020
	10	2314	2017	2	5	N3KFAA	flyU_flights_2020
	11	1096	2017	2	8	N3AAAA	flyU_flights_2020

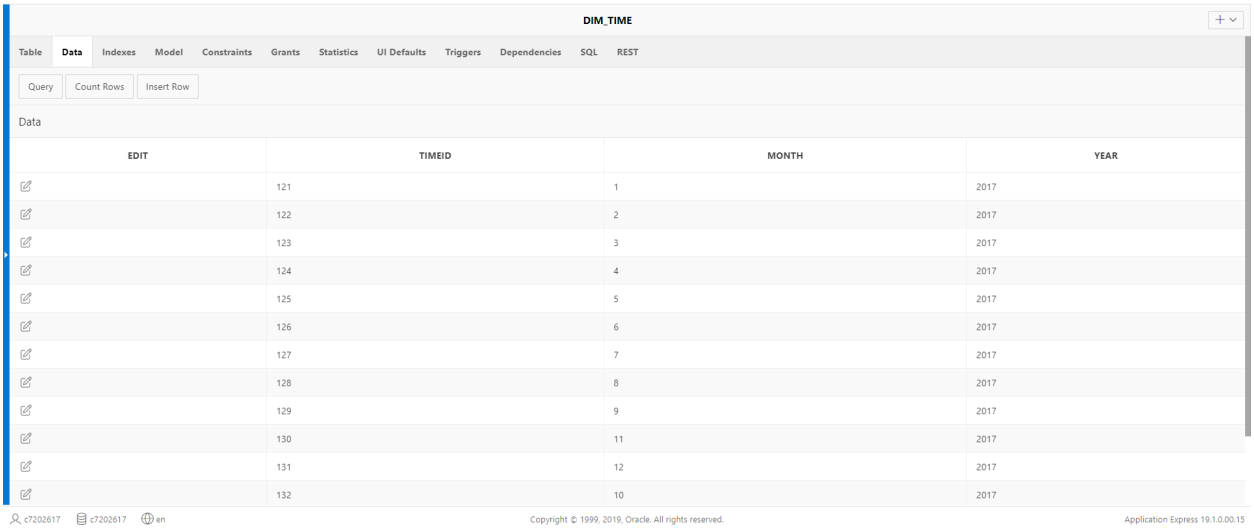
LOAD:

In this part of the ETL process, we load all the data from the transformation table into the star schema design we upload above in task 1.



These are the dimension table with quality data that can be analyzed:

- **DIM_TIME**



- **DIM_FLIGHTS**

DIM_FLIGHTS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Query | Count Rows | Insert Row

Data

EDIT	FLIGHTKEY	FLIGHTNO	TAILNO	DBSOURCE
	1	291	N3GYAA	flyU_flights_2020
	2	2314	N3LCAA	flyU_flights_2020
	3	1096	N3DJAA	flyU_flights_2020
	4	291	N3KHAA	flyU_flights_2020
	5	2314	N3DJAA	flyU_flights_2020
	6	2314	N3JYAA	flyU_flights_2020
	7	291	N3GPAA	flyU_flights_2020
	8	1096	N3CMAA	flyU_flights_2020
	9	2314	N3CMAA	flyU_flights_2020
	10	2314	N3KFAA	flyU_flights_2020
	11	1096	N3AAAA	flyU_flights_2020
	12	1096	N3ADAA	flyU_flights_2020

- **DIM_COMPLAINTS**

DIM_COMPLAINTS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Query | Count Rows | Insert Row

Data

EDIT	COMPLAINTKEY	COMPLAINTID	COMPLAINTTYPE	DESCRIPTION	COMPLAINTSTATUS	DBSOURCE
	61	61	L	late	open	flyU_flights_2020
	62	62	L	late	open	flyU_flights_2020
	63	63	L	late	open	flyU_flights_2020
	64	64	L	late	open	flyU_flights_2020
	65	65	L	late	open	flyU_flights_2020
	66	66	L	late	open	flyU_flights_2020
	67	67	L	late	open	flyU_flights_2020
	68	68	L	late	open	flyU_flights_2020
	69	84	L	late	open	flyU_flights_2020
	70	85	A	unknown	open	flyU_flights_2020
	71	86	C	cancelled	closed	flyU_flights_2020
	72	87	C	cancelled	closed	flyU_flights_2020

c7202617 c7202617 en

Application Express 19.1.0.00.15

- **DIM_COMPENSATION**

DIM_COMPENSATION

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST

Query | Count Rows | Insert Row

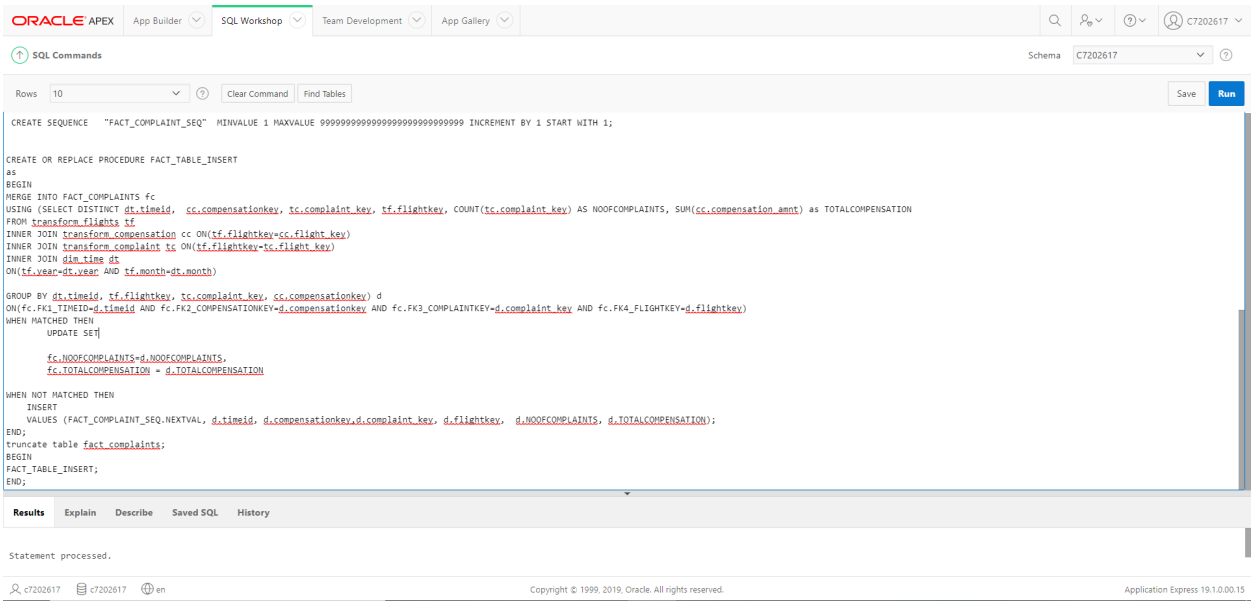
Data

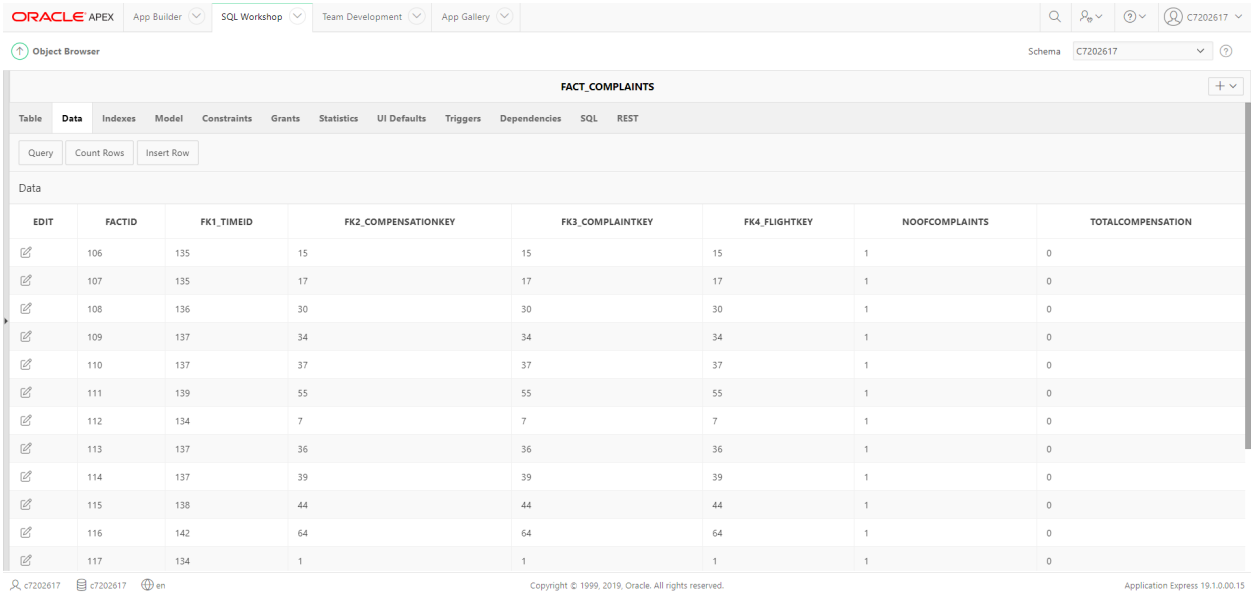
EDIT	COMPENSATIONKEY	COMPENSATION_AMNT	COMPENSATION_TYPE	DBSOURCE
	1	0	0	flyU_flights_2020
	2	0	0	flyU_flights_2020
	3	0	0	flyU_flights_2020
	4	0	0	flyU_flights_2020
	5	0	0	flyU_flights_2020
	6	0	0	flyU_flights_2020
	7	0	0	flyU_flights_2020
	8	0	0	flyU_flights_2020
	9	0	0	flyU_flights_2020
	10	0	0	flyU_flights_2020
	11	0	0	flyU_flights_2020
	12	0	0	flyU_flights_2020

c7202617 c7202617 en

Application Express 19.1.0.00.15

- **FACT_COMPLAINTS**

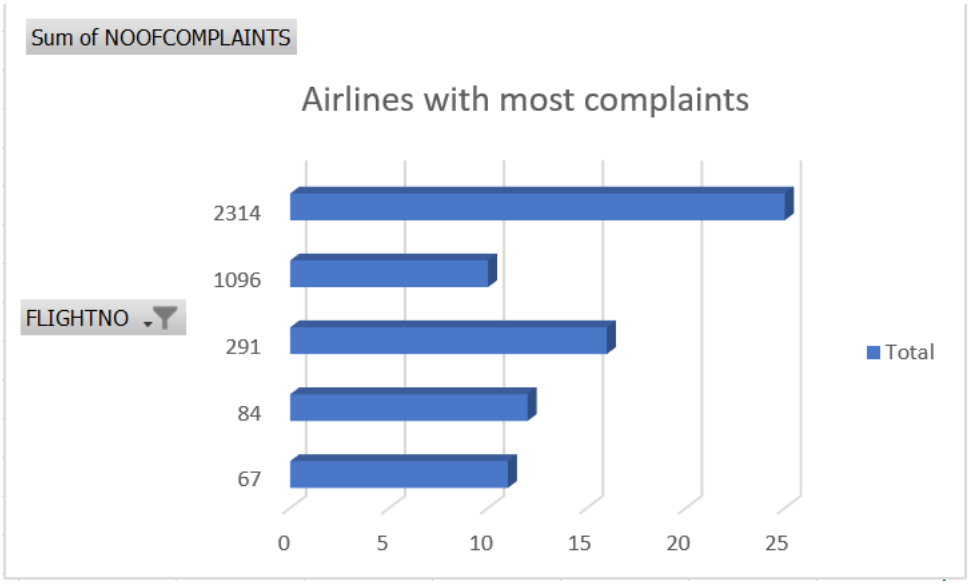




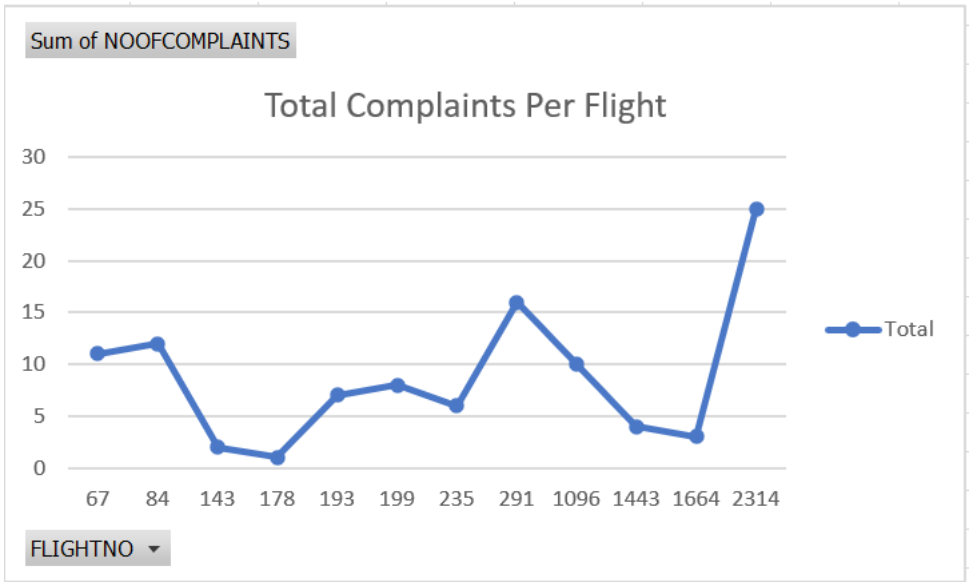
Task 4

Data Analysis:

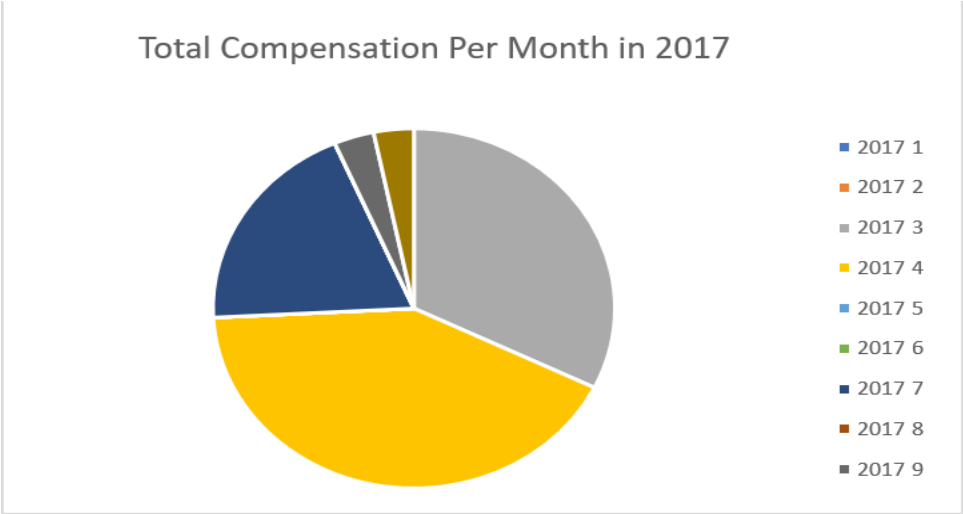
- Report 1



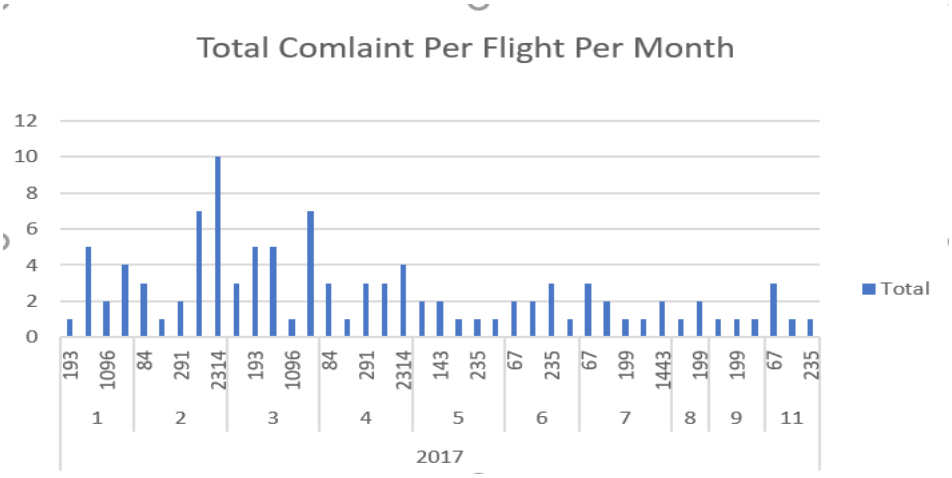
- Report 2



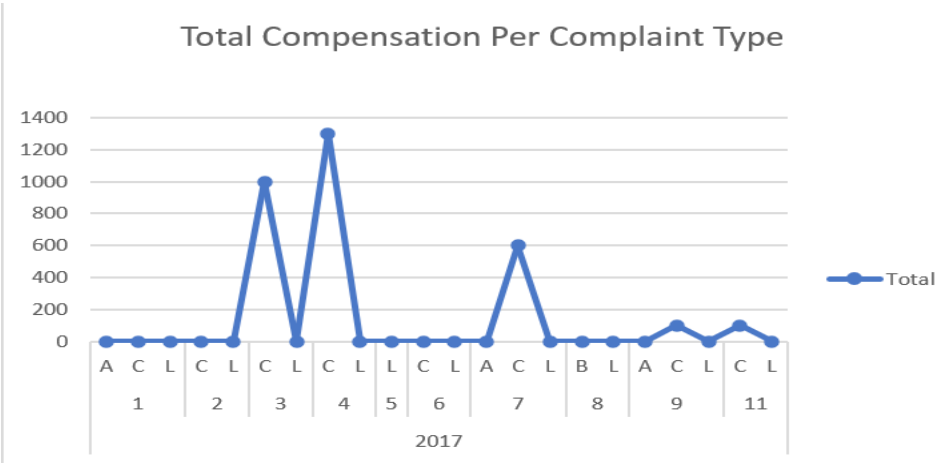
- Report 3



- Report 4

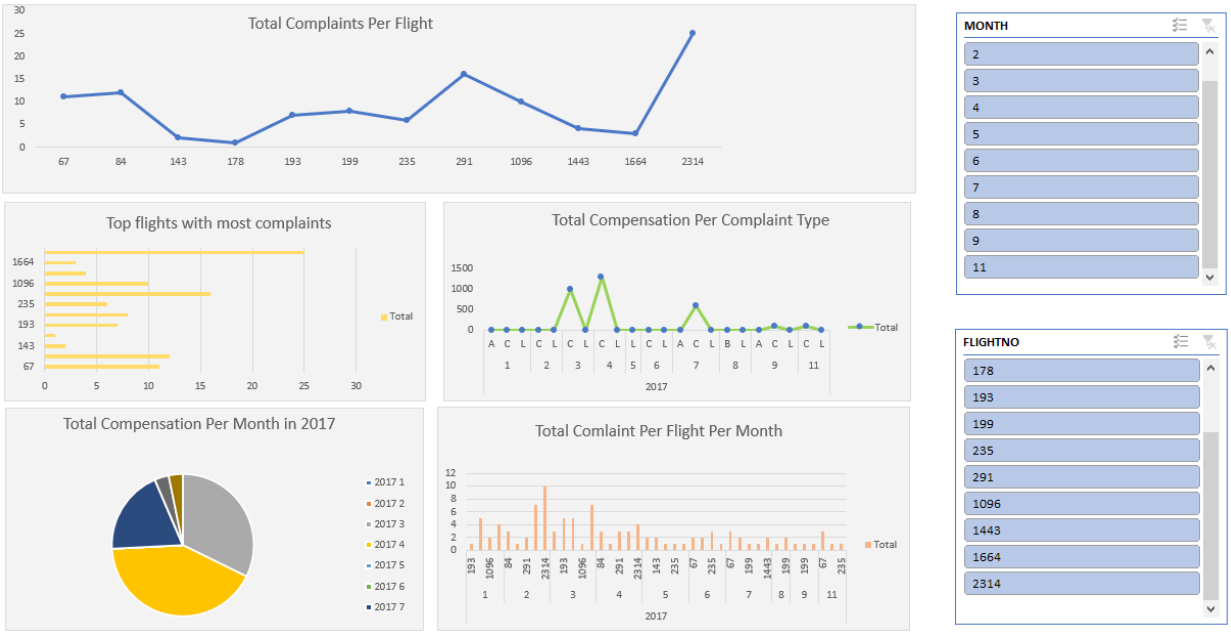


- Report 5



Dashboard:

This dashboard facilitates the company to make multi-dimensional analysis based on flights, complaints and time.



Task 5

Data Warehouse Approaches

In Data Warehouse, there are two major approaches when it comes to designing and they are:

- Inmon Method
- Kimball Method

When it comes to effective corporate performance, it is crucial to determine the appropriate approach according to the requirement of the project. This helps cut down project cost as well as save a lot of time. Both methods have their own advantages and differentiating factors, so determining which method to use determines the future of the company.

Bill Inmon's Method

The Bill Inmon's architecture, also known as the top-down design, organizes data using ER modelling. In this architecture, a normalised data model is designed before the dimensional data marts where all the required data are created from the data warehouse. The method enacts data warehouse as a centralised repository where it captures the "atomic" data at the very lowest level of detail hence, earning the name, atomic data warehouse. Therefore, it provides a logical framework for delivering business intelligence as it is at the centre of the corporate information factory (CIF). Simply, it is starting with building a big, centralized enterprise data warehouse where all available data from transaction systems are consolidated into a subject-oriented, integrated, time-variant and non-volatile collection of data that supports decision making. Then data marts are built for analytic needs of departments.

The Inmon design approach uses the normalized form for building entity structure, avoiding data redundancy as much as possible. This results in clear identification of business requirements and improving any data irregularities.

Advantages of Bill Immon's Method:

The Inmon architecture offers the following advantages :

- The data warehouse acts as a centralized unit for the entire company, where data from multiple sources can be integrated.
- This approach data warehouse process is less likely to result failure as it avoids data redundancy as much as possible resulting in relatively less data irregularities.
- As the top-down model represents data at a very lowest level of detail, making decision making and analytical process simpler.
- This approach is greatly flexible, as it is easier to update the data warehouse in case there is any change in the data sources , time, business requirements, etc.
- It can handle diverse enterprise-wide reporting requirements.

Disadvantages of Bill Immon's Method:

The Inmon architecture offers the following disadvantages :

- It can be susceptible to more complexity because over time, multiple tables are added to the data warehouse.
- It can be expensive in terms of hiring resources skilled in data science.
- The initial setup and delivery can take a lot of time.

- Additional ETL operation is required since data marts are created after the creation of the data warehouse.

Ralph Kimball Method

The Ralph Kimball architecture, also known as bottom-up design of Data Warehouse(DW), forms data marts first based on the business requirements. In this architecture, the key business questions and the key business processes are identified before the primary data sources are evaluated. Once, the data sources are analysed and documented, the Extract, Transform and Load (ETL) software is utilized to fetch data from multiple sources and load into a staging area. After that, the data purification process occurs where data are segregated into clean and error table. The data in error table are then recleaned and transformed into the clean data. From here, data is loaded into a dimensional which is not normalized. The dimensional modelling is done using the star schema. In the star schema, there is typically a fact table surrounded by many dimensions. The fact table has all the measures that are relevant to the subject area, and it also has the foreign keys from the different dimensions that surround the fact. The dimensions are denormalized completely so that the user can drill up and drill down without joining to another table.

The Kimball design approach uses the denormalized form for building entity structure. It is also based on conformed facts i.e. data marts which are separately implemented are grouped together with a robust architecture.

Advantages of Bill Immon's Method:

The Inmon architecture offers the following advantages :

- The initial setup and execution is faster as there is no normalization involved.
- It simplifies querying and analysis as the data operators can be easily interpreted because of its denormalized structure.
- It takes less space in the database which makes system management simpler.
- A smaller team of designers and planners is sufficient for data warehouse management.
- It provides multi-dimensional structure and helps generate reliable insights.

Disadvantages of Bill Immon's Method:

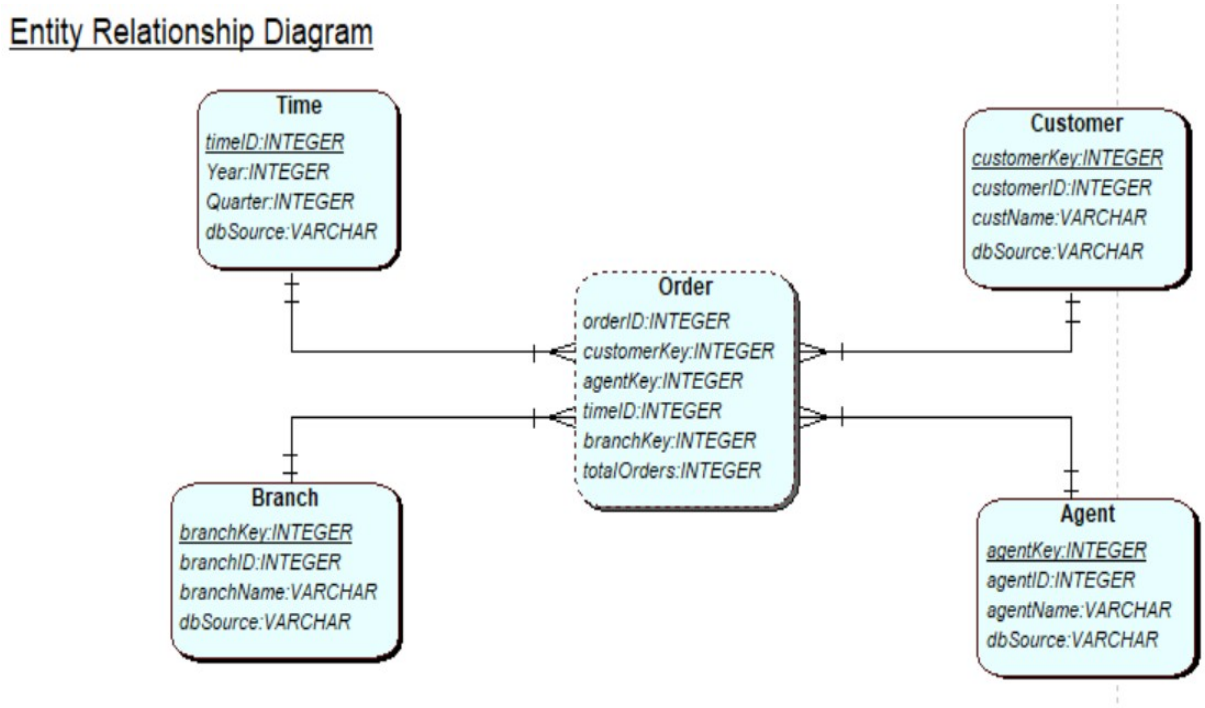
The Inmon architecture offers the following disadvantages :

- In Kimball design, data isn't entirely integrated before reporting.
- As redundant data is added to database tables, data irregularities are most likely to occur.
- In the Kimball DW approach, the data warehouse model may be difficult to alter with any change in business needs.
- The model is business process-oriented so it won't focus on the other areas of the enterprise.

Assignment Portfolio

Data Warehouse design for LBU business:

- Design the star schema for the DW to be implemented.



- Define, fact table, dimension(s), attributes, keys and measures.

  Fact: Order Table (OrderID, customerKey, agentKey, timeID, branchKey, totalOrders)

Dimensions: Customer (customerKey, customerID, custName, dbSource)

Agent (agentKey, agentID, agentName, dbSource)

Branch (branchKey, branchID, branchName, dbSource)

Time (timeID, Year, Quarter, dbSource)

Measures: totalOrders

- Reports:

a. Number of orders made in UK first quarter of the year, in comparison with last year.

```
SELECT t.Quarter, SUM (totalOrders)
FROM Order o, Time t
WHERE o.timeID = t.timeID
AND t.Quarter = "Q1"
AND TO_CHAR(SYSDATE, 'YYYY') = t.Year
GROUP BY  t.Quarter
UNION
SELECT t.Quarter, SUM (totalOrders)
FROM Order o, Time t
WHERE o.timeID = t.timeID
AND TO_CHAR(SYSDATE, 'YYYY') -1 = t.Year
GROUP BY  t.Quarter'
```

b. Who is our best customer, in first quarter of this year?

```
SELECT SUM (totalOrders), customerKey
FROM
(
SELECT SUM (totalOrders), customerKey
RANK OVER (ORDER BY SUM (totalOrders) DESC) AS Rank
```

FROM Order o, Time t,
WHERE o.timeID = t.timeID
AND t.Quarter = “Q1”
AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year
)
WHERE Rank <=1;
GROUP BY customerKey;

c. Total number of orders made in first quarter of the year, in comparison with last year for each branch?

SELECT t.Quarter, SUM (totalOrders), branchKey
FROM Order o, Time t
WHERE o.timeID = t.timeID
AND t.Quarter = “Q1”
AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year
GROUP BY t.Quarter, branchKey
UNION
SELECT t.Quarter, SUM (totalOrders), branchKey
FROM Order o, Time t
WHERE o.timeID = t.timeID
AND TO_CHAR(SYSDATE, ‘YYYY’) -1 = t.Year
GROUP BY t.Quarter, branchKey

d. Who is our best Agent, in the first quarter of this year?

SELECT SUM (totalOrders), agentKey
FROM
(
SELECT SUM (totalOrders), agentKey
RANK OVER (**ORDER BY** SUM (totalOrders) **DESC**) **AS** Rank
FROM Order o, Time t,
WHERE o.timeID = t.timeID

AND t.Quarter = “Q1”

AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year

)

WHERE Rank <=1;

GROUP BY agentKeY

Data Warehouse design for a wholesale furniture company

1. Identify facts, dimensions and measures:

Fact: Sales

Measures: Quantity, Income, Discount

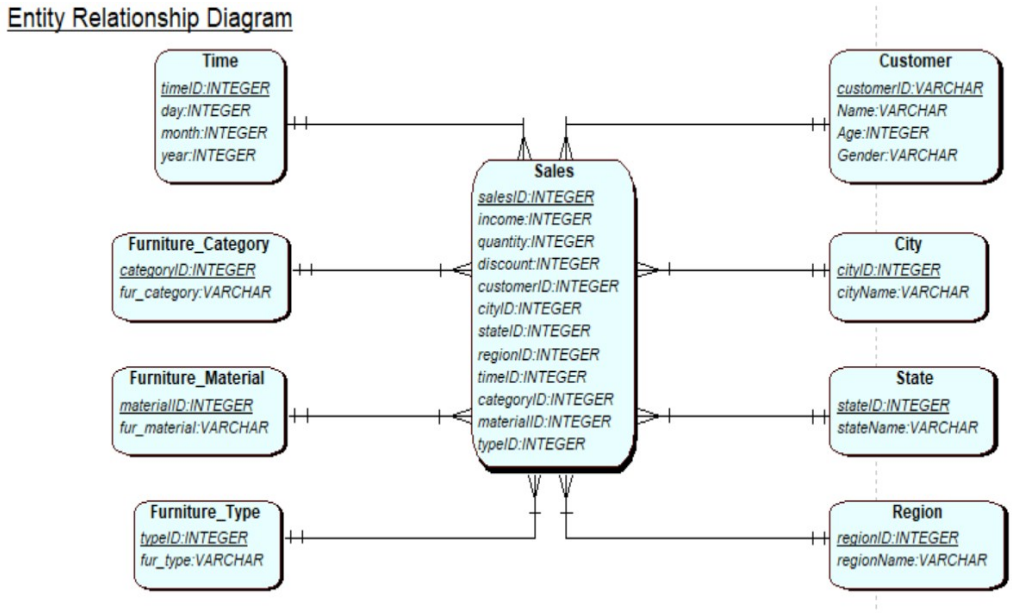
Dimension: Furniture (Type, Category, Material)

Customer (Age, Gender,

2. For each fact:

- Produce the fact schema

STAR SCHEMA



a. Find the quantity, the total income and discount with respect to each city, type of furniture and the month

```
SELECT SUM(quantity), SUM(income), SUM(discount), cityID, typeID, timeID
FROM Sales
GROUP BY cityID, typeID, timeID
```

b. Find the average quantity, income and discount with respect to each country, furniture material and year

```
SELECT AVG(quantity), AVG(income), AVG(discount), stateID, materialID, timeID
FROM Sales
GROUP BY stateID, materialID, timeID
```

c. Determine the 5 most sold furniture during the May month

```
SELECT typeID, SUM(quantity) as Total
FROM (
SELECT typeID, SUM(quantity) as Total,
RANK() OVER (ORDER BY SUM(quantity) DESC) as rank
FROM Sales s, Time t
WHERE  t.month = “May”
)
WHERE rank <= 5
```