LEEDS BECKETT UNIVERSITY

The British College

DATA WAREHOUSE

-Prabhat Adhikari

Table of Contents

Background

The project is based off an aircraft company named FlyU. They hold records identified with their flights, for example, take-off times, appearance times, customers on each flight, number of trips to arrive at their destination, return flights, tail number, grievances raised, and any sort of flight delays.

The company stores utilizing Oracle just as Excel sheets. They handle this everyday information to design, deal with their client grumblings support and convey a quality assistance.

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 essential to comprehend what is data warehouse. In this way, in straightforward terms, it is a kind of data management system that stores an company information(data) from at least one sources. The fundamental reason for data warehouse is to think about and investigate information for more noteworthy corporate execution. With regards to business knowledge, it is viewed as one the imperative parts as it helps uphold business choices by giving analytical techniques and a more extensive understanding into the data warehouse. It is intended to execute query and analysis on historical data derived from heterogeneous sources multiple sources.

PURPOSE OF DW:

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:

Data Warehousing - OLAP. Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information.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 Transactional Processing) is a category of data processing that is focused on transaction-oriented tasks. OLTP typically involves inserting, updating, and/or deleting small amounts of data in a database.

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 includes planning, budgeting, forecasting, and analysis	OLTP is useful to administer day to 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

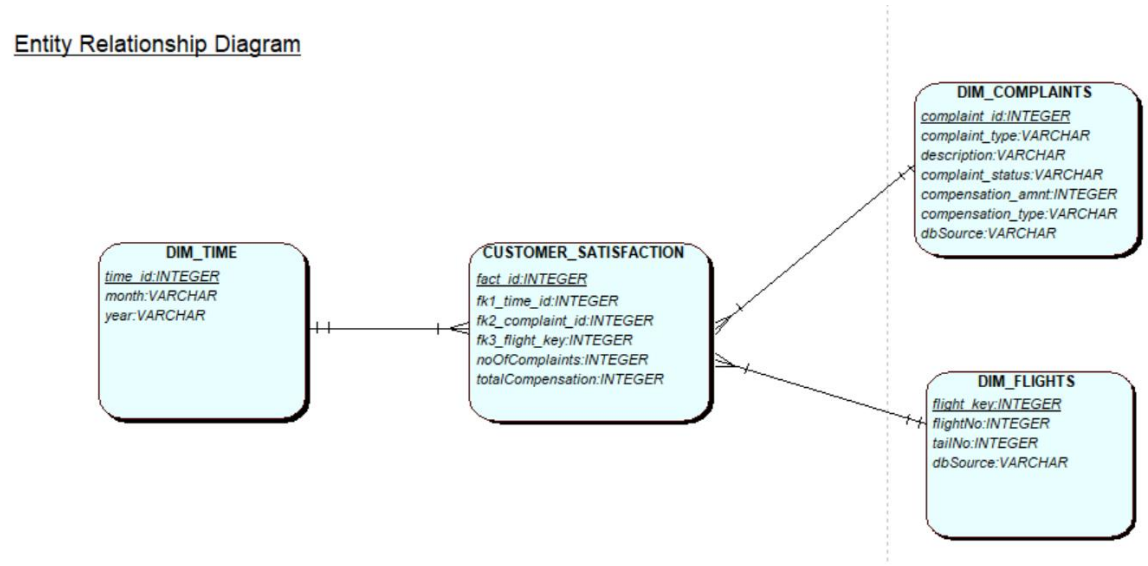
KPI 2: Ensure customer satisfaction

Task 1

Reports:

- Identify the no of customer complaints per complaint type per flight registered in the year 2017.
- The number of complains per month.
- Total amount of compensation given per month.
- Total amount of compensation given per month per flight.
- Total number of complaints per month per complaint type.

STAR SCHEMA:



Star schema is one of the crucial components of Data Warehouse. In the star schema, there is a fact table “Customer Satisfaction” surrounded by many dimensions “Dim Flights”, “Dim Complaints” and “Dim Time”. The fact table has the foreign keys from the different dimensions as well as all the measures that are relevant to the project reports.

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
	quarter	VARCHAR	No	Should be generated	n/a	n/a	Create a sequence quarter_seq to generate quarterly dates
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_COMPLAINTS

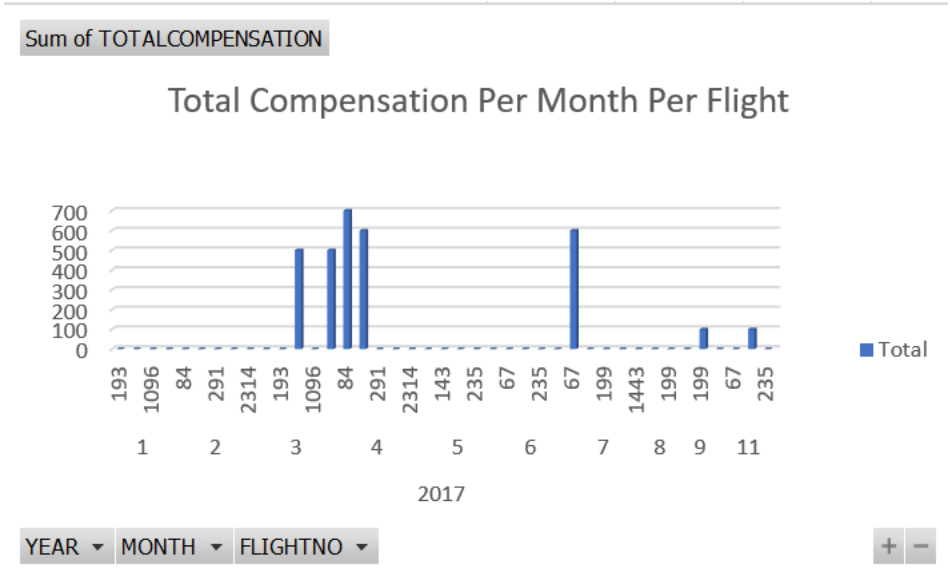
Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
DIM_ COMPLAINTS	Complaint Key	INTEGER	Yes	Automatically generated as surrogate key	n/a	n/a	Create a sequence complain_seq to generate primary keys
	Complaint Id	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	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
	compensation_amnt	INTEGER	No	FlyU_flights	n/a	n/a	n/a
	compensation_type	VARCHAR	No	FlyU_flights	Null value	Some complaints are missing the compensation type	Will need to mention the compensation type
	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_CUSTOMER_SATISFACTION

Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
▪ FACT_CUSTOMER_SATISFACTION	FactId	INTEGER	Yes	Automatically generated as primary key	n/a	n/a	Create a sequence 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
	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:							

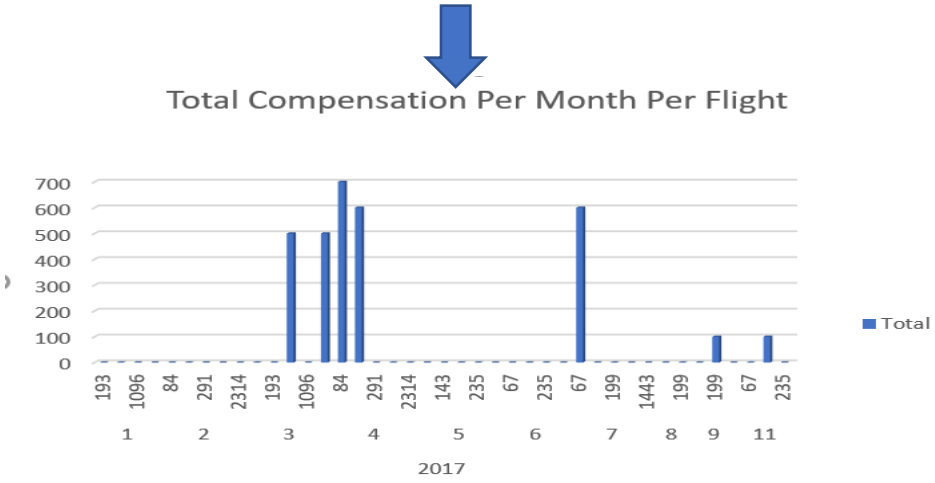
ANALYTICAL REPORT:

- EXCEL REPORT



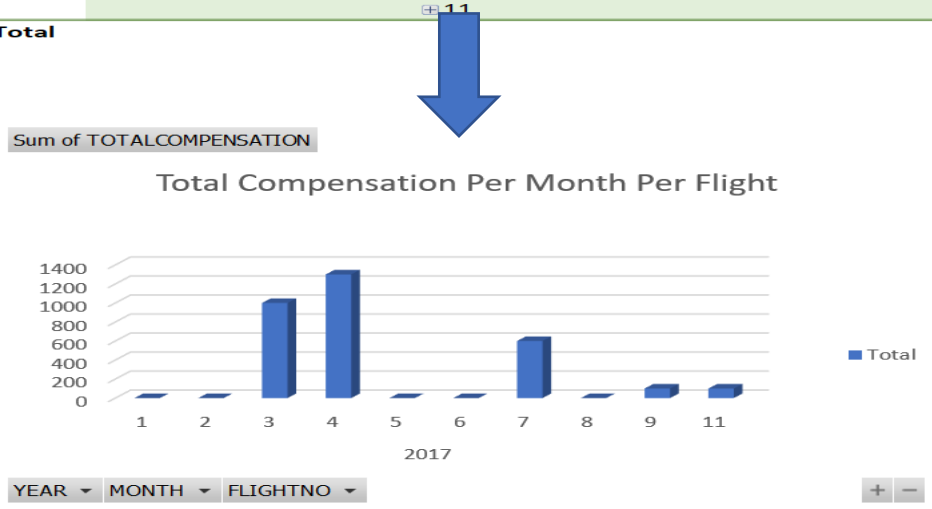
- DRILL DOWN

YEAR	MONTH	FLIGHTNO	Sum of TOTALCOMPENSATION
2017			3100
	1		0
		193	0
		291	0
		1096	0
		2314	0
	2		0
		84	0
		193	0
		291	0
		1096	0
		2314	0
	3		1000
		84	0
		193	0
		291	500
		1096	0
		2314	500
	4		1300

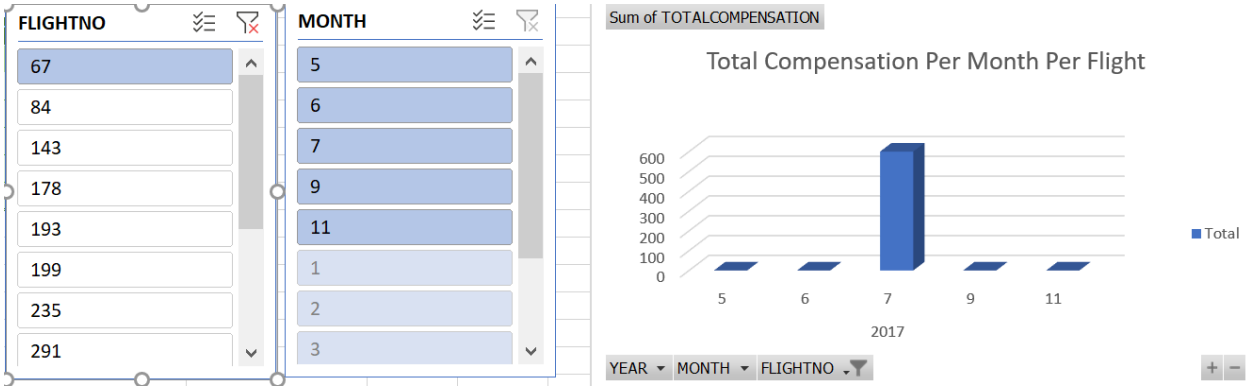


- DRILL UP

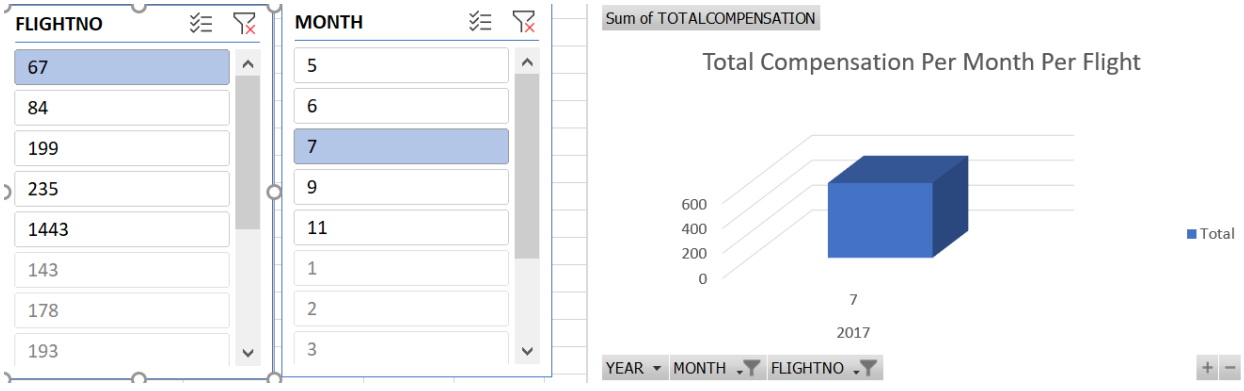
YEAR	MONTH	FLIGHTNO	Sum of TOTALCOMPENSATION
2017			3100
	1		0
	2		0
	3		1000
	4		1300
	5		0
	6		0
	7		600
	8		0
	9		100
	11		100
Grand Total			3100



- SLICING



- DICING

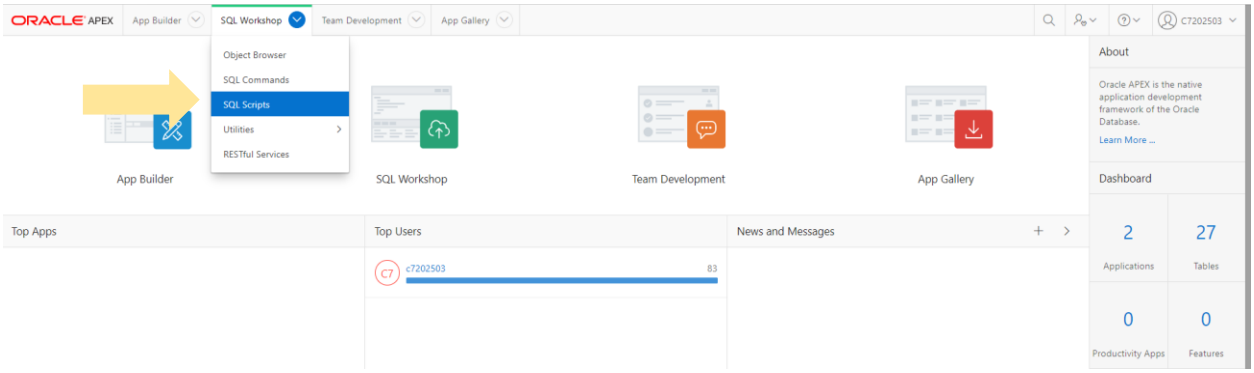


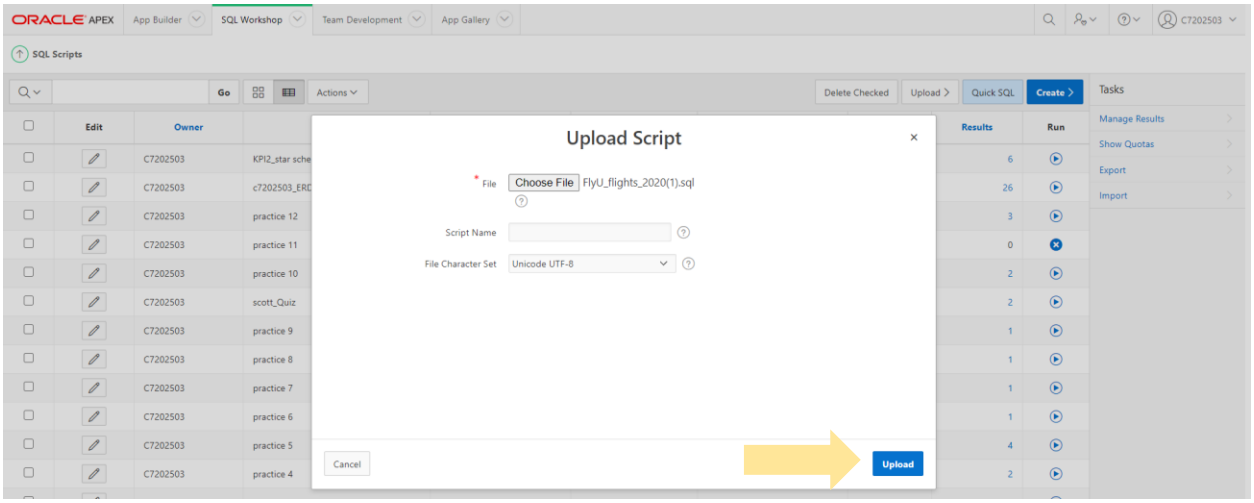
Task 2

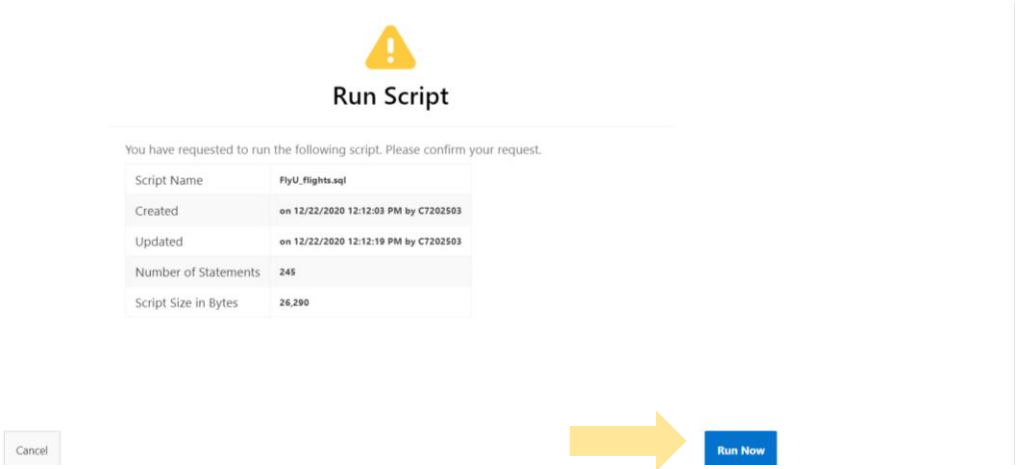
STAGING AREA SETUP:

In the staging area setup, we have extracted data from the data source: flyU_flights as we only need data related to the customer satisfaction.

The pictures below demonstrate the uploading process of the data source.







Number	Elapsed	Statement	Feedback	Rows
1	0.02	DROP TABLE FlyU_flights CASCADE CONSTRAINTS	Table dropped.	0
2	0.01	DROP TABLE customer CASCADE CONSTRAINTS	Table dropped.	0
3	0.02	DROP TABLE complaint CASCADE CONSTRAINTS	Table dropped.	0
4	0.01	CREATE TABLE customer(customer_id INTEGER NOT NULL, custo	Table created.	0
5	0.01	CREATE TABLE complaint(complaint_id INTEGER NOT NULL PRIMA	Table created.	0
6	0.01	CREATE TABLE FlyU_flights (flight_the_year NUMBER(4), t	Table created.	0
7	0.01	ALTER TABLE FlyU_flights ADD CONSTRAINT pk1_flights PRIMARY	Table altered.	0
8	0.01	ALTER TABLE complaint ADD CONSTRAINT fk1_complaint_to_custom	Table altered.	0
9	0.01	ALTER TABLE complaint ADD CONSTRAINT fk1_complaint_to_flight	Table altered.	0
10	0.01	INSERT INTO Customer VALUES (10, 'NY10', 'BUSINESS', 'IBM',	1 row(s) inserted.	1
11	0.00	INSERT INTO Customer VALUES (101, 'NY101', 'BUSINESS', 'Goog	1 row(s) inserted.	1
12	0.00	INSERT INTO Customer VALUES (102, 'NY102', 'BUSINESS', 'Amaz	1 row(s) inserted.	1
13	0.00	INSERT INTO Customer VALUES (103, 'NY103', 'BUSINESS', 'Face	1 row(s) inserted.	1
14	0.01	INSERT INTO Customer VALUES (104, 'NY104', 'BUSINESS', 'HP',	1 row(s) inserted.	1
15	0.00	INSERT INTO Customer VALUES (105, 'NY105', 'BUSINESS', 'Sky8	1 row(s) inserted.	1

Download

row(s) 1 - 15 of 245 Next ▶

245 Statements Processed

245 Successful

0 With Errors

- FlyU_flights

FLYU_FLIGHTS

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
FLIGHT_THE_YEAR	NUMBER(4,0)	No	-	1
THE_MONTH	NUMBER(2,0)	No	-	2
THE_DAY	NUMBER(2,0)	No	-	3
FLIGHT_NUMBER	NUMBER(10,0)	No	-	4
D_O_W	NUMBER(1,0)	Yes	-	-
TAIL_NUMBER	VARCHAR2(10)	Yes	-	-
ORIGIN_AIRPORT	VARCHAR2(6)	Yes	-	-
DESTINATION_AIRPORT	VARCHAR2(6)	Yes	-	-
ARRIVAL_DELAY	VARCHAR2(10)	Yes	-	-
DIVERTED	NUMBER(1,0)	Yes	-	-
CANCELLED	NUMBER(1,0)	Yes	-	-
CANCELLED_REASON	VARCHAR2(1)	Yes	-	-

Download | Print

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	193	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

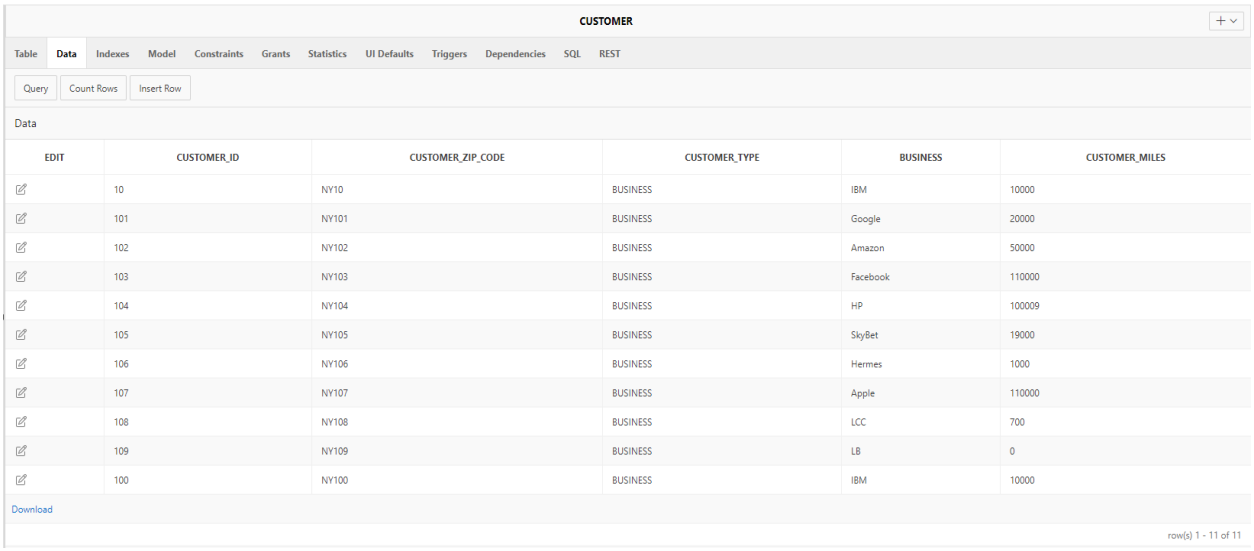
CUSTOMER

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

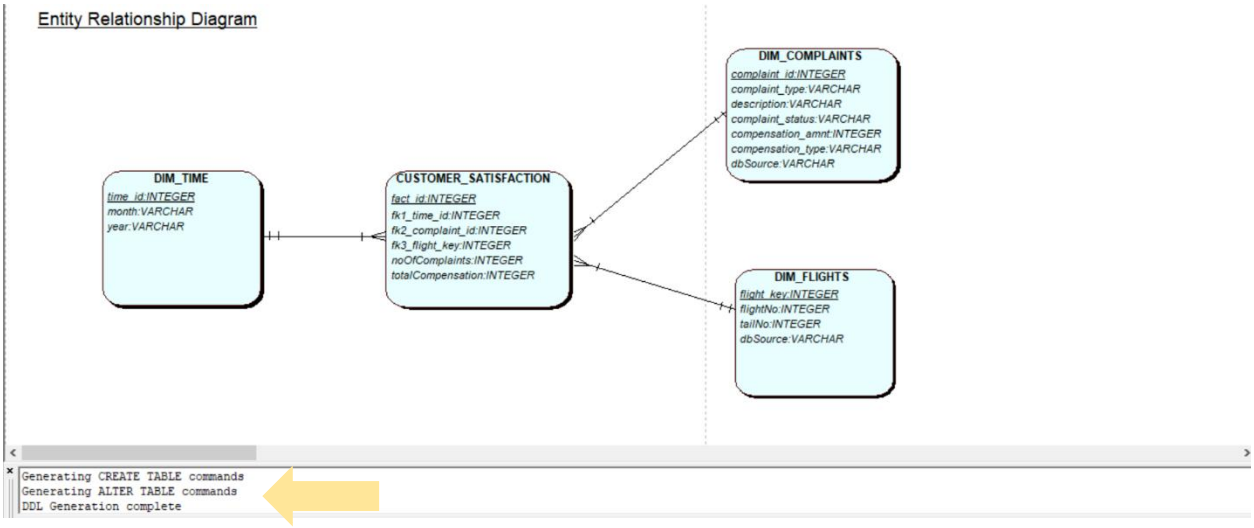
EDIT	CUSTOMER_ID	CUSTOMER_ZIP_CODE	CUSTOMER_TYPE	BUSINESS	CUSTOMER_MILES
	10	NY10	BUSINESS	IBM	10000
	101	NY101	BUSINESS	Google	20000
	102	NY102	BUSINESS	Amazon	50000
	103	NY103	BUSINESS	Facebook	110000
	104	NY104	BUSINESS	HP	100009
	105	NY105	BUSINESS	SkyBet	19000
	106	NY106	BUSINESS	Hermes	1000
	107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

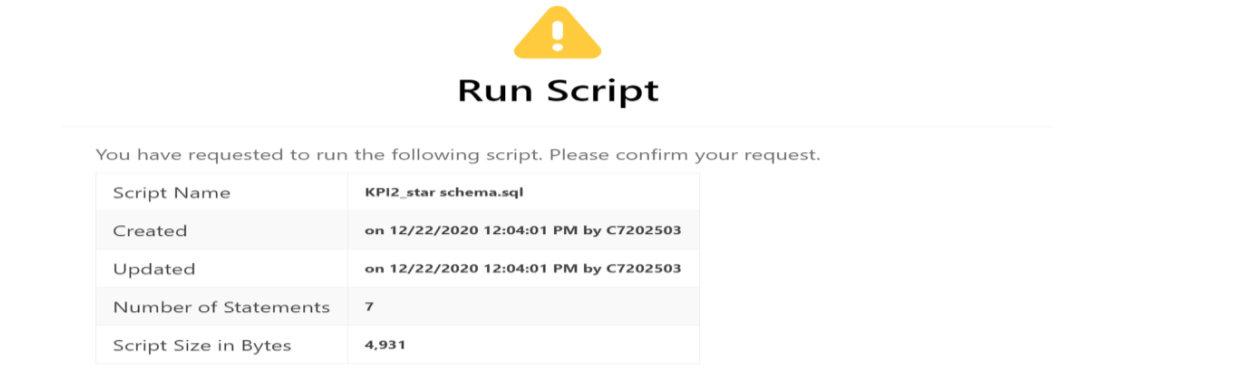
Download

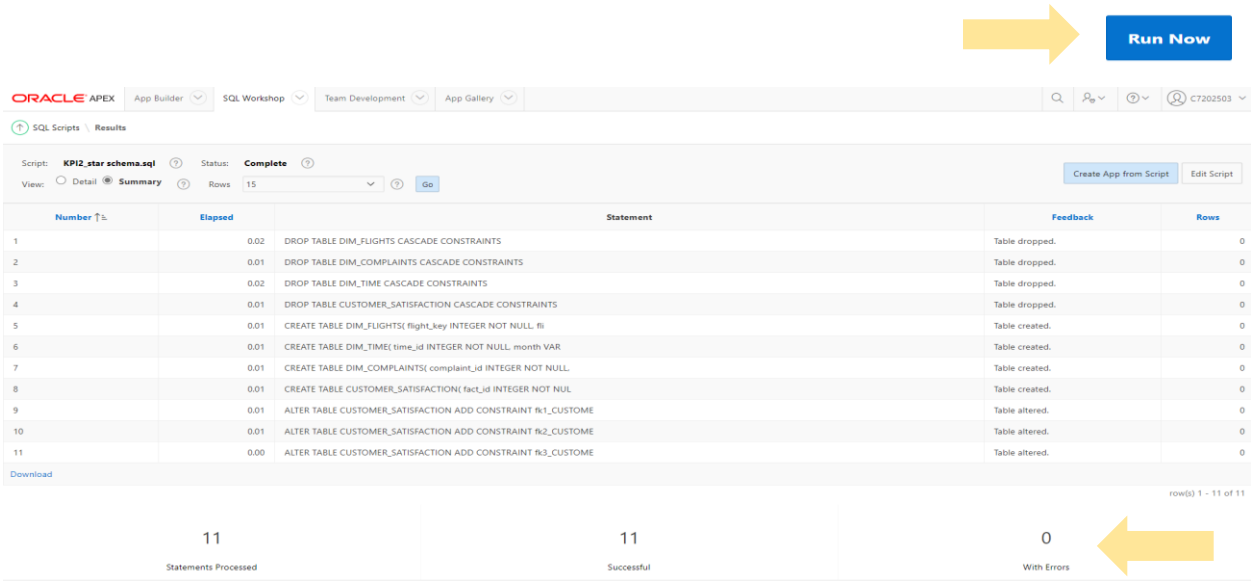
row(s) 1 - 11 of 11

STAR SCHEMA SETUP:

The star schema that has been generated is converted into a sql script and uploaded into database using the following steps:







Task 3

ETL(Extraction, Transformation, Load)

ETL is short for extract, transform, load. It is an important data warehouse(DW) tool to pull data from multiple database and place it into another database. ETL is one of the key tool for today's business intelligence (BI) processes and systems. It is the process which facilitates analytical tools and important business insights.

EXTRACTION:

Extraction is the process of migrating data from multiple and different types of sources. In this case, we have already extracted the data, now we are going to transfer the data into a staging area.

Firstly, the stage tables are created along with their primary keys, foreign keys, attributes, etc. There are two staging tables created based on our star schema design and they are:

Stage Complaints:

```
-----------------------------------------
DROP TABLE STG_COMPLAINTS CASCADE CONSTRAINTS;

CREATE STAGING COMPLAINTS TABLE
=========================================
CREATE TABLE STG_COMPLAINTS
(
complaint_id INTEGER NOT NULL PRIMARY KEY,
complaint_type VARCHAR2(5),
description VARCHAR2(20),
complaint_status VARCHAR2(7),
compensation_amnt NUMBER(*,0),
compensation_type VARCHAR2(8),
flight_key INTEGER NOT NULL,
dbSource VARCHAR(17)
);

ALTER TABLE  STG_COMPLAINTS ADD CONSTRAINT "FK1_STG_COMPLAINT_TO_FLIGHT" FOREIGN KEY (flight_key)
          REFERENCES  STG_FLIGHTS(flight_key);
```

Stage Flights:

```
-----------------------------------------
DROP TABLE STG_COMPLAINTS CASCADE CONSTRAINTS;

CREATE STAGING COMPLAINTS TABLE
=========================================
CREATE TABLE STG_COMPLAINTS
(
complaint_id INTEGER NOT NULL PRIMARY KEY,
complaint_type VARCHAR2(5),
description VARCHAR2(20),
complaint_status VARCHAR2(7),
compensation_amnt NUMBER(*,0),
compensation_type VARCHAR2(8),
flight_key INTEGER NOT NULL,
dbSource VARCHAR(17)
);

ALTER TABLE  STG_COMPLAINTS ADD CONSTRAINT "FK1_STG_COMPLAINT_TO_FLIGHT" FOREIGN KEY (flight_key)
          REFERENCES  STG_FLIGHTS(flight_key);
```

Data Insertion:

In the figure below, the staging is done using STG_DATA_INSERT procedure which uses loop to insert data into all the staging tables. This ensures that the data are accurate as it is sent based on the foreign key.

```
INSERT DATA INTO STAGING TABLES
================================================
create or replace procedure STG_DATA_INSERT
as
next NUMBER;
begin

        FOR i IN (SELECT * FROM FlyU_Flights) LOOP
                next := STG_FLIGHT_SEQ.nextval;
                INSERT INTO STG_FLIGHTS(flight_key, flight_no, year, month, day, tailNo, dbSource)
                VALUES(next, 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 STG_COMPLAINTS(complaint_id, complaint_type, description, complaint_status, compensation_amnt, compensation_type, flight_Key, dbSource)
               VALUES(j.complaint_id, j.complaint_type, j.description, j.complaint_status, j.compensation_amnt, j.compensation_type, next,
               'flyU_flights_2020');


        END LOOP;

    END LOOP;
END;
begin
STG_DATA_INSERT;
end;
```

After that, the tables look like following:

STG_COMPLAINTS

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

Query | Count Rows | Insert Row

Data

EDIT	COMPLAINT_ID	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	1	-	late	open	0	0	1	flyU_flights_2020
	2	-	late	open	0	0	2	flyU_flights_2020
	3	-	late	open	0	0	3	flyU_flights_2020
	4	-	late	open	0	0	4	flyU_flights_2020
	5	-	late	open	0	0	5	flyU_flights_2020
	6	-	late	open	0	0	6	flyU_flights_2020
	7	-	late	open	0	0	7	flyU_flights_2020

STG_FLIGHTS

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

Query | Count Rows | Insert Row

Data

EDIT	FLIGHT_KEY	FLIGHT_NO	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

TRANSFORMATION:

Transform is the process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database.

In this part of the ETL process, we identify all the bad and good data 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 TRANSFORMATION

Firstly, two tables where clean and error data will be segregated are created.

```
CREATING CLEAN DATA TABLE FOR COMPLAINTS TABLE
==============================================
truncate table clean_complaints;
DROP table clean_complaints CASCADE CONSTRAINTS;
create table clean_complaints
(
        "COMPLAINT_ID" NUMBER(*,0) NOT NULL PRIMARY KEY,
        "COMPLAINT_TYPE" VARCHAR2(5),
        "DESCRIPTION" VARCHAR2(20),
        "COMPLAINT_STATUS" VARCHAR2(7),
        "COMPENSATION_AMNT" NUMBER(*,0),
        "COMPENSATION_TYPE" VARCHAR2(8),
        "FLIGHT_KEY" INTEGER NOT NULL,
        "DBSOURCE" VARCHAR2(17)
);
```

```
CREATING ERROR DATA TABLE FOR COMPLAINTS TABLE
==============================================
truncate table error_complaints;
DROP table error_complaints CASCADE CONSTRAINTS;
create table error_complaints
(
        "ERROR_ID" NUMBER(*,0) NOT NULL PRIMARY KEY,
        "COMPLAINT_ID" NUMBER(*,0) NOT NULL ENABLE,
        "COMPLAINT_TYPE" VARCHAR2(5),
        "DESCRIPTION" VARCHAR2(20),
        "COMPLAINT_STATUS" VARCHAR2(7),
        "COMPENSATION_AMNT" NUMBER(*,0),
        "COMPENSATION_TYPE" VARCHAR2(8),
        "FLIGHT_KEY" INTEGER NOT NULL,
        "DBSOURCE" VARCHAR2(17),
        "ERROR_DESCRIPTION" VARCHAR2(15),
        "STATUS" VARCHAR2(10),
        "RESOLUTION_DATE" DATE
);
```

Now, that the tables are created, we are going to identify the clear and error data using procedure and populate the tables.

```
create or replace procedure check_complaints_data(no_rows OUT NUMBER)
IS
BEGIN
no_rows := 0;

INSERT INTO error_complaints
(
SELECT  ERROR_COMPLAINTS_SEQ.nextval, c.COMPLAINT_ID, c.COMPLAINT_TYPE, c.DESCRIPTION, c.COMPLAINT_STATUS, c.COMPENSATION_AMNT, c.COMPENSATION_TYPE, c.FLIGHT_KEY, c.DBSOURCE, 'error val', 'not fixed', sysdate
FROM STG_COMPLAINTS c
WHERE COMPLAINT_TYPE IS NULL
OR DESCRIPTION IS NULL
OR COMPLAINT_STATUS IS NULL
OR COMPENSATION_AMNT IS NULL
OR COMPENSATION_TYPE IS NULL
OR description ='cancelled' AND COMPLAINT_TYPE <>'C'
);
no_rows := TO_CHAR(SQL%RowCount);

DELETE FROM clean_complaints;
INSERT INTO clean_complaints
(
SELECT * FROM STG_COMPLAINTS
WHERE COMPLAINT_TYPE IS NOT NULL
AND DESCRIPTION IS NOT NULL
AND COMPLAINT_STATUS IS NOT NULL
AND COMPENSATION_AMNT IS NOT NULL
AND COMPENSATION_TYPE IS NOT NULL
AND description ='cancelled' AND COMPLAINT_TYPE ='C'
);
END;
```

Then, the bad table looks as shown in the picture. Here, all the bad data will have error description, status, and resolution date.

ERROR_COMPLAINTS

Data

EDIT	ERROR_ID	COMPLAINT_ID	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE	ERROR_DESCRIPTION	STATUS	RESOLUTION_DATE
	1047	1	-	late	open	0	0	1	flyU_flights_2020	error val	not fixed	12/22/2020
	1048	2	-	late	open	0	0	2	flyU_flights_2020	error val	not fixed	12/22/2020
	1049	3	-	late	open	0	0	3	flyU_flights_2020	error val	not fixed	12/22/2020
	1050	4	-	late	open	0	0	4	flyU_flights_2020	error val	not fixed	12/22/2020
	1051	5	-	late	open	0	0	5	flyU_flights_2020	error val	not fixed	12/22/2020
	1052	6	-	late	open	0	0	6	flyU_flights_2020	error val	not fixed	12/22/2020
	1053	7	-	late	open	0	0	7	flyU_flights_2020	error val	not fixed	12/22/2020
	1054	8	-	late	open	0	0	8	flyU_flights_2020	error val	not fixed	12/22/2020
	1055	9	-	late	open	0	0	9	flyU_flights_2020	error val	not fixed	12/22/2020
	1056	10	-	late	open	0	0	10	flyU_flights_2020	error val	not fixed	12/22/2020

As we have identified the bad data, we will now reclean the error data and make it useable.

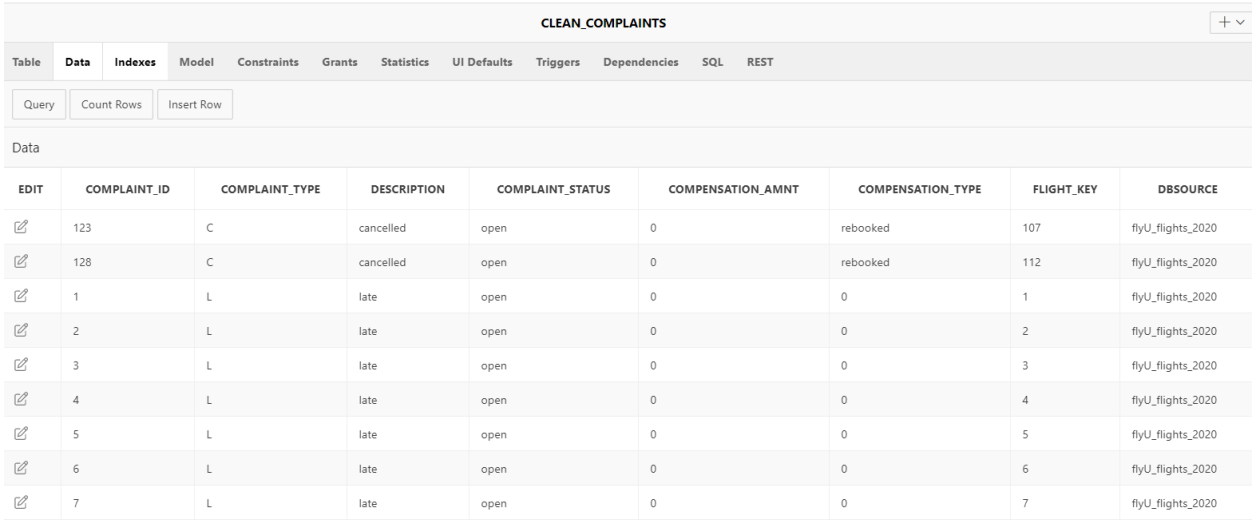
This is done by using a procedure as shown below.

```
Reclean bad data in Complaint error table
==========================================

create or replace procedure reclean_complaint_bad_data(no_rows OUT NUMBER)
IS
BEGIN
no_rows := 0;

        update error_complaints set DESCRIPTION ='unknown', STATUS ='fixed', RESOLUTION_DATE = sysdate where DESCRIPTION is null;
        no_rows := TO_CHAR(SQL%RowCount);

        update error_complaints set COMPLAINT_TYPE='L', STATUS ='fixed', RESOLUTION_DATE = sysdate where COMPLAINT_TYPE is null and DESCRIPTION ='late';
        no_rows := no_rows + TO_CHAR(SQL%RowCount);

        update error_complaints set COMPLAINT_TYPE='C', STATUS ='fixed', RESOLUTION_DATE = sysdate  where (COMPLAINT_TYPE='A' OR COMPLAINT_TYPE='B') and DESCRIPTION ='cancelled';
        no_rows := no_rows + TO_CHAR(SQL%RowCount);

         update error_complaints set COMPENSATION_TYPE ='unknown', STATUS ='fixed', RESOLUTION_DATE = sysdate where COMPENSATION_TYPE is null;
        no_rows := no_rows + TO_CHAR(SQL%RowCount);
end;

DECLARE
noOfRows NUMBER(5,2);
begin
reclean_complaint_bad_data(noOfRows);
DBMS_OUTPUT.PUT_LINE('No of bad data cleaned: '||noOfRows);
END;
```

After recleaning, since the bad and good data are segregated, we will now bring them in the same table using a merge statement.

```
create or replace procedure RECLEANED_COMPLAINT_DATA_MERGE
as
begin
MERGE INTO clean_complaints C
 USING error_complaints B
 ON (C.COMPLAINT_ID = B.COMPLAINT_ID)
WHEN MATCHED THEN
 UPDATE SET

C.COMPLAINT_TYPE = B.COMPLAINT_TYPE,
C.DESCRIPTION=B.DESCRIPTION,
C.COMPLAINT_STATUS=B.COMPLAINT_STATUS,
C.COMPENSATION_AMNT = B.COMPENSATION_AMNT,
C.COMPENSATION_TYPE = B.COMPENSATION_TYPE,
C.FLIGHT_KEY = B.FLIGHT_KEY,
C.DBSOURCE=B.DBSOURCE

WHEN NOT MATCHED THEN
  INSERT VALUES
(B.COMPLAINT_ID, B.COMPLAINT_TYPE, B.DESCRIPTION, B.COMPLAINT_STATUS, B.COMPENSATION_AMNT,B.COMPENSATION_TYPE, B.FLIGHT_KEY, B.DBSOURCE);
END;

begin
RECLEANED_COMPLAINT_DATA_MERGE;
end;
```

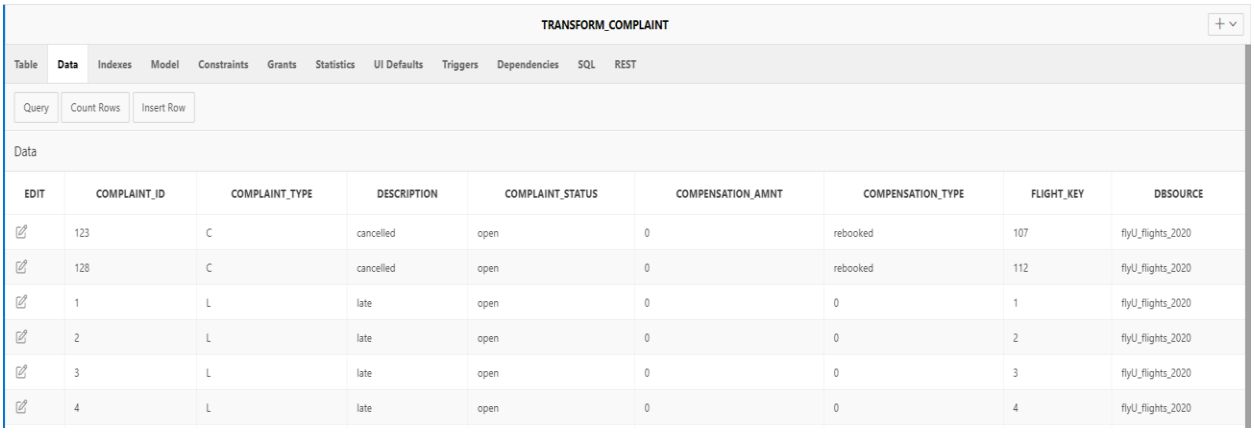
The figure below shows the reclean data as well as the clean data from before.



EDIT	COMPLAINT_ID	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	123	C	cancelled	open	0	rebooked	107	flyU_flights_2020
	128	C	cancelled	open	0	rebooked	112	flyU_flights_2020
	1	L	late	open	0	0	1	flyU_flights_2020
	2	L	late	open	0	0	2	flyU_flights_2020
	3	L	late	open	0	0	3	flyU_flights_2020
	4	L	late	open	0	0	4	flyU_flights_2020
	5	L	late	open	0	0	5	flyU_flights_2020
	6	L	late	open	0	0	6	flyU_flights_2020
	7	L	late	open	0	0	7	flyU_flights_2020

Finally, we will move the data from the cleaning stage to the transformation table so that it can be loaded into the dimensional model.

```
--TRANSFORM_COMPLAINT--
DROP TABLE TRANSFORM_COMPLAINT CASCADE CONSTRAINTS;
TRUNCATE TABLE TRANSFORM_COMPLAINT;
CREATE TABLE  TRANSFORM_COMPLAINT
(

COMPLAINT_ID NUMBER NOT NULL PRIMARY KEY,
"COMPLAINT_TYPE" VARCHAR2(5),
"DESCRIPTION" VARCHAR2(200),
"COMPLAINT_STATUS" VARCHAR2(7),
"COMPENSATION_AMNT" NUMBER,
"COMPENSATION_TYPE" VARCHAR2(20),
"FLIGHT_KEY" INTEGER NOT NULL,
"DBSOURCE" VARCHAR2(17)
) ;

--PROCEDURE TO TRANSFORM GOOD COMPLAINT TABLE
create or replace procedure TRANSFORM_COMPLAINT_TABLE
as
begin
DELETE FROM TRANSFORM_COMPLAINT;
INSERT INTO TRANSFORM_COMPLAINT (SELECT * FROM CLEAN_COMPLAINTS);
END;

begin
TRANSFORM_COMPLAINT_TABLE;
end;
```



EDIT	COMPLAINT_ID	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	123	C	cancelled	open	0	rebooked	107	flyU_flights_2020
	128	C	cancelled	open	0	rebooked	112	flyU_flights_2020
	1	L	late	open	0	0	1	flyU_flights_2020
	2	L	late	open	0	0	2	flyU_flights_2020
	3	L	late	open	0	0	3	flyU_flights_2020
	4	L	late	open	0	0	4	flyU_flights_2020

FLIGHTS TABLE TRANSFORMATION

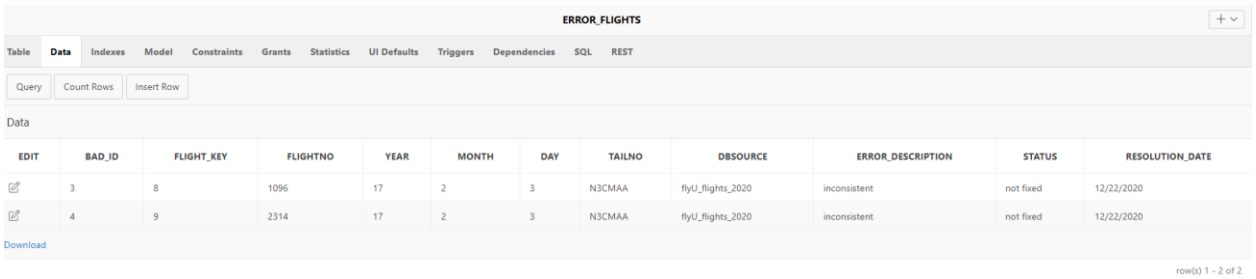
Firstly, two tables where clean and error data will be segregated are created.

```
CREATING BAD DATA TABLE FOR FLIGHTS TABLE
==========================================
DROP table error_flights CASCADE CONSTRAINTS;
create table error_flights
(
"BAD_ID" NUMBER(*,0) NOT NULL PRIMARY KEY,
        "FLIGHT_KEY" NUMBER(*,0),
        "FLIGHTNO" NUMBER(10,0),
        "YEAR" NUMBER(4,0),
        "MONTH" NUMBER(2,0),
        "DAY" NUMBER(2,0),
        "TAILNO" VARCHAR2(10),
        "DBSOURCE" VARCHAR2(17),
        "ERROR_DESCRIPTION" VARCHAR2(15),
        "STATUS" VARCHAR2(10),
        "RESOLUTION_DATE" DATE
);
============================================
CREATING GOOD DATA TABLE FOR FLIGHTS TABLE
============================================
DROP table clean_flights CASCADE CONSTRAINTS;
create table clean_flights
(
FLIGHT_KEY NUMBER NOT NULL PRIMARY KEY,
"FLIGHTNO" NUMBER(10,0),
        "YEAR" NUMBER(4,0),
        "MONTH" NUMBER(2,0),
        "DAY" NUMBER(2,0),
        "TAILNO" VARCHAR2(10),
        "DBSOURCE" VARCHAR2(17)
);
============================================
```

Now, that the tables are created, we are going to identify the clear and error data using procedure, and populate the tables.

```
create or replace procedure check_flights_data(no_rows OUT NUMBER)
IS
BEGIN
no_rows := 0;

INSERT INTO error_flights
(
SELECT ERROR_FLIGHTS_SEQ.nextval, F.FLIGHT_KEY, F.FLIGHT_NO, F.YEAR, F.MONTH, F.DAY, F.TAILNO, F.DBSOURCE, 'null val', 'not fixed', sysdate
FROM STG_FLIGHTS F
WHERE FLIGHT_NO IS NULL
OR YEAR IS NULL
OR MONTH IS NULL
OR DAY IS NULL
OR TAILNO IS NULL);
no_rows := TO_CHAR(SQL%RowCount);

INSERT INTO error_flights
(
SELECT ERROR_FLIGHTS_SEQ.nextval, F.FLIGHT_KEY, F.FLIGHT_NO, F.YEAR, F.MONTH, F.DAY, F.TAILNO, F.DBSOURCE, 'inconsistent', 'not fixed', sysdate
FROM STG_FLIGHTS F
WHERE F.YEAR='17');
no_rows := no_rows + TO_CHAR(SQL%RowCount);

DELETE FROM clean_flights;
INSERT INTO clean_flights
(
SELECT * FROM STG_FLIGHTS
WHERE FLIGHT_NO IS NOT NULL
OR YEAR IS NOT NULL
OR MONTH IS NOT NULL
OR DAY IS NOT NULL
OR TAILNO IS NOT NULL
AND YEAR <>'17');
END;
```

Then, the bad table looks as shown in the picture. Here, all the bad data will have error description, status, and resolution date.



As we have identified the bad data, we will now reclean the error data and make it useable.

This is done by using a procedure as shown below.

```
-----------------------------------------
Reclean bad data in FLIGHT bad table--
-----------------------------------------

create or replace procedure reclean_flight_bad_data(pv_rows OUT NUMBER)
IS
BEGIN
pv_rows := 0;


        update error_flights set YEAR='2017', STATUS ='fixed', RESOLUTION_DATE = sysdate where YEAR='17';
        pv_rows := pv_rows + TO_CHAR(SQL%RowCount);

end;

-----------------------------------------
EXECUTE Compensation Table PROCEDURE
-----------------------------------------
DECLARE
noOfRows NUMBER(5,2);
begin
reclean_flight_bad_data(noOfRows);
DBMS_OUTPUT.PUT_LINE('No of bad data cleaned: '||noOfRows);
END;
```

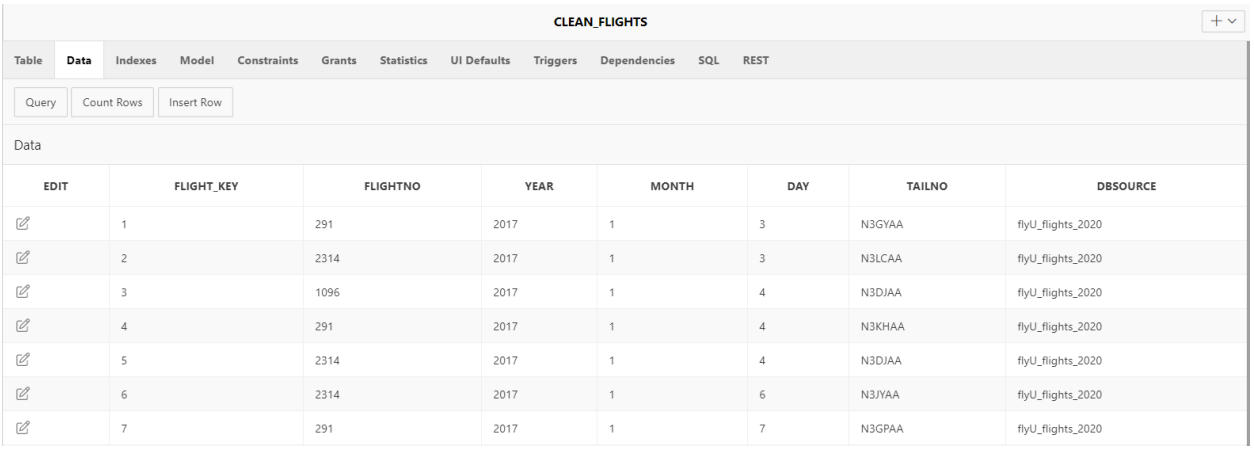
Results | Explain | Describe | Saved SQL | History

```
No of bad data cleaned: 2

Statement processed.
```

Since the bad and good data are segregated, we will now bring them in the same table using a merge statement.

```
create or replace procedure RECLEANED_FLIGHT_DATA_MERGE
as
begin
MERGE INTO clean_flights C
 USING error_flights B
 ON (C.flight_key = B.flight_key)
WHEN MATCHED THEN
 UPDATE SET
C.FLIGHTNO = B.FLIGHTNO,
C.YEAR = B.YEAR,
C.MONTH = B.MONTH,
C.DAY = B.DAY,
C.TAILNO=B.TAILNO,
C.DBSOURCE=B.DBSOURCE

WHEN NOT MATCHED THEN
  INSERT VALUES
(B.flight_key, B.FLIGHTNO, B.YEAR, B.MONTH, B.DAY, B.TAILNO, B.DBSOURCE);
END;

begin
RECLEANED_FLIGHT_DATA_MERGE;
end;
```

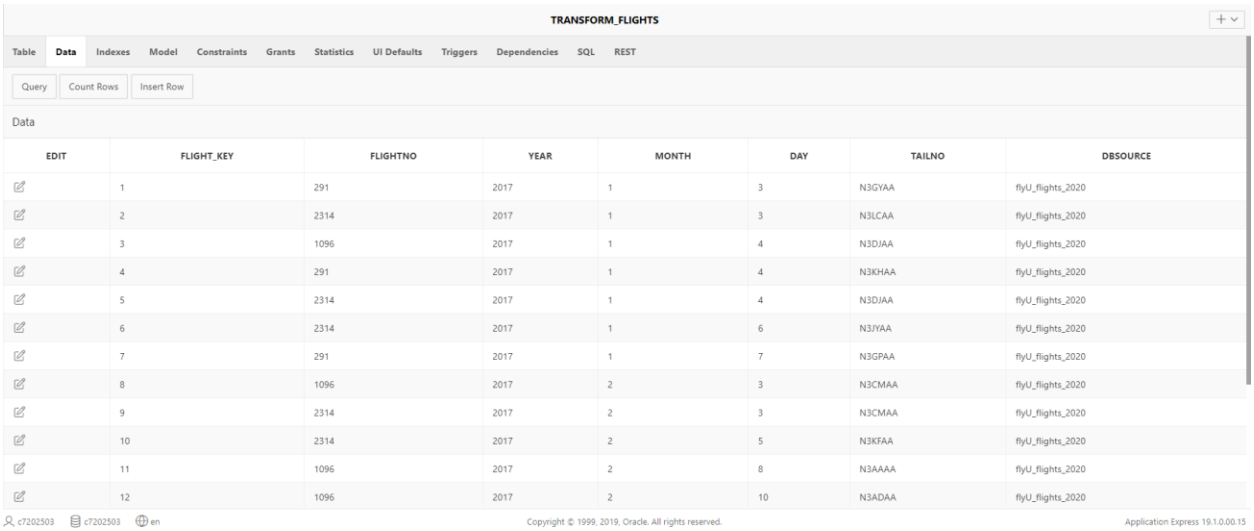
The figure below shows the reclean data as well as the clean data from before.



EDIT	FLIGHT_KEY	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

Finally, we will move the data from the cleaning stage to the transformation table so that it can be loaded into the dimensional model.

```
--TRANSFORM_FLIGHTS--
DROP TABLE TRANSFORM_FLIGHTS CASCADE CONSTRAINTS;
TRUNCATE TABLE TRANSFORM_FLIGHTS;
CREATE TABLE  TRANSFORM_FLIGHTS
(
         FLIGHT_KEY NUMBER NOT NULL PRIMARY KEY,
        "FLIGHTNO" NUMBER(10,0),
        "YEAR" NUMBER(4,0),
        "MONTH" NUMBER(2,0),
        "DAY" NUMBER(2,0),
        "TAILNO" VARCHAR2(10),
        "DBSOURCE" VARCHAR2(17)
) ;

--PROCEDURE TO TRANSFORM GOOD FLIGHTS TABLE
create or replace procedure TRANSFORM_FLIGHTS_TABLE
as
begin
DELETE FROM TRANSFORM_FLIGHTS;
INSERT INTO TRANSFORM_FLIGHTS (SELECT * FROM CLEAN_FLIGHTS);
END;

begin
TRANSFORM_FLIGHTS_TABLE;
end;
```



EDIT	FLIGHT_KEY	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
	12	1096	2017	2	10	N3ADAA	flyU_flights_2020

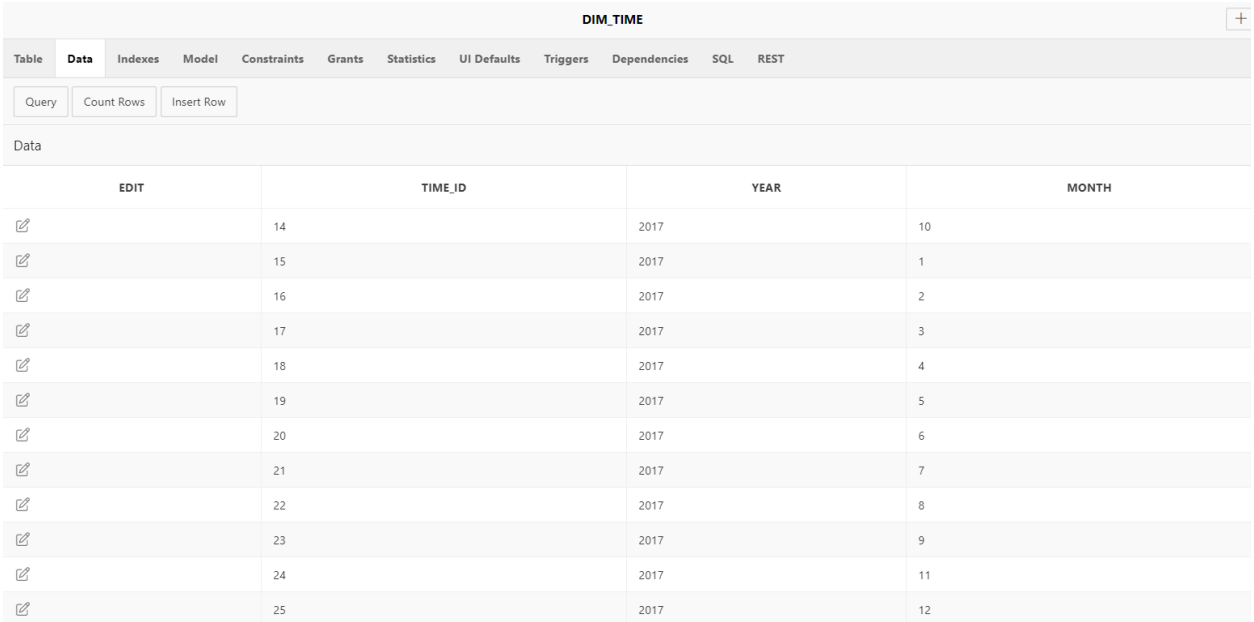
LOAD:

Load is the process of writing the data into the target database. 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.

The insertion is done using procedure which utilizes loop to insert data with unique constraints into the dimension tables.

```
CREATE OR REPLACE PROCEDURE DIM_TABLE_INSERT
as
       v_count NUMBER;
BEGIN
    FOR i IN (SELECT * FROM transform_flights) LOOP
        INSERT INTO dim_flights(flight_key, flightno, tailno, dbsource) VALUES(i.flight_key, i.flightno, i.tailno, i.dbsource);

        SELECT count(*) INTO v_count FROM dim_time WHERE year=i.year and month=i.month;

        IF v_count=0   then
        INSERT INTO dim_time(time_id, year, month) VALUES(DIM_TIME_SEQ.nextval, i.year, i.month);
        end if;
        FOR j IN (SELECT * FROM transform_complaint WHERE flight_key=i.flight_key) LOOP
               INSERT INTO dim_complaints(COMPLAINT_ID, COMPLAINT_TYPE, DESCRIPTION, COMPLAINT_STATUS, COMPENSATION_AMNT, COMPENSATION_TYPE, DBSOURCE)
              VALUES( j.COMPLAINT_ID, j.COMPLAINT_TYPE, j.DESCRIPTION, j.COMPLAINT_STATUS, j.COMPENSATION_AMNT, j.COMPENSATION_TYPE, j.DBSOURCE );
        END LOOP;

    END LOOP;
END;

INSERT INTO dim_time(time_id, year, month) VALUES(DIM_TIME_SEQ.nextval,2017, 10);
BEGIN
DIM_TABLE_INSERT;
END;
```

- 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	FLIGHT_KEY	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	COMPLAINT_ID	COMPLAINT_TYPE	DESCRIPTION	COMPLAINT_STATUS	COMPENSATION_AMNT	COMPENSATION_TYPE	DBSOURCE
	1	L	late	open	0	0	flyU_flights_2020
	2	L	late	open	0	0	flyU_flights_2020
	3	L	late	open	0	0	flyU_flights_2020
	4	L	late	open	0	0	flyU_flights_2020
	5	L	late	open	0	0	flyU_flights_2020
	6	L	late	open	0	0	flyU_flights_2020
	7	L	late	open	0	0	flyU_flights_2020
	8	L	late	open	0	0	flyU_flights_2020
	9	L	late	open	0	0	flyU_flights_2020
	10	L	late	open	0	0	flyU_flights_2020
	11	L	late	open	0	0	flyU_flights_2020
	12	L	late	open	0	0	flyU_flights_2020

- CUSTOMER SATISFACTION (FACT TABLE)

The data insertion in fact table is done using procedure which utilizes the merge statement to insert data with unique constraints into the dimension tables.

```
 CREATE SEQUENCE   "FACT_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE PROCEDURE FACT_TABLE_INSERT
as
BEGIN
MERGE INTO CUSTOMER_SATISFACTION cs
USING (SELECT t.time_id, c.complaint_ID, f.flight_key, COUNT(c.complaint_id) AS NOOFCOMPLAINTS, SUM(c.compensation_amnt) as TOTALCOMPENSATION
FROM transform_flights f
JOIN transform_complaint c ON(f.flight_key=c.flight_key)
JOIN dim_time t ON(f.year=t.year AND f.month = t.month)

GROUP BY t.time_id, f.flight_key, c.complaint_id) d
ON(cs.FK1_TIME_ID=d.time_id AND cs.FK2_FLIGHT_KEY=d.flight_key AND cs.FK3_COMPLAINT_ID=d.complaint_id)
WHEN MATCHED THEN
        UPDATE SET

        cs.NOOFCOMPLAINTS=d.NOOFCOMPLAINTS,
        cs.TOTALCOMPENSATION = d.TOTALCOMPENSATION

WHEN NOT MATCHED THEN
    INSERT
    VALUES (FACT_SEQ.NEXTVAL, d.time_id, d.flight_key, d.complaint_id,  d.NOOFCOMPLAINTS, d.TOTALCOMPENSATION);
END;

BEGIN
FACT_TABLE_INSERT;
END;
```

CUSTOMER_SATISFACTION

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

Query | Count Rows | Insert Row

Data

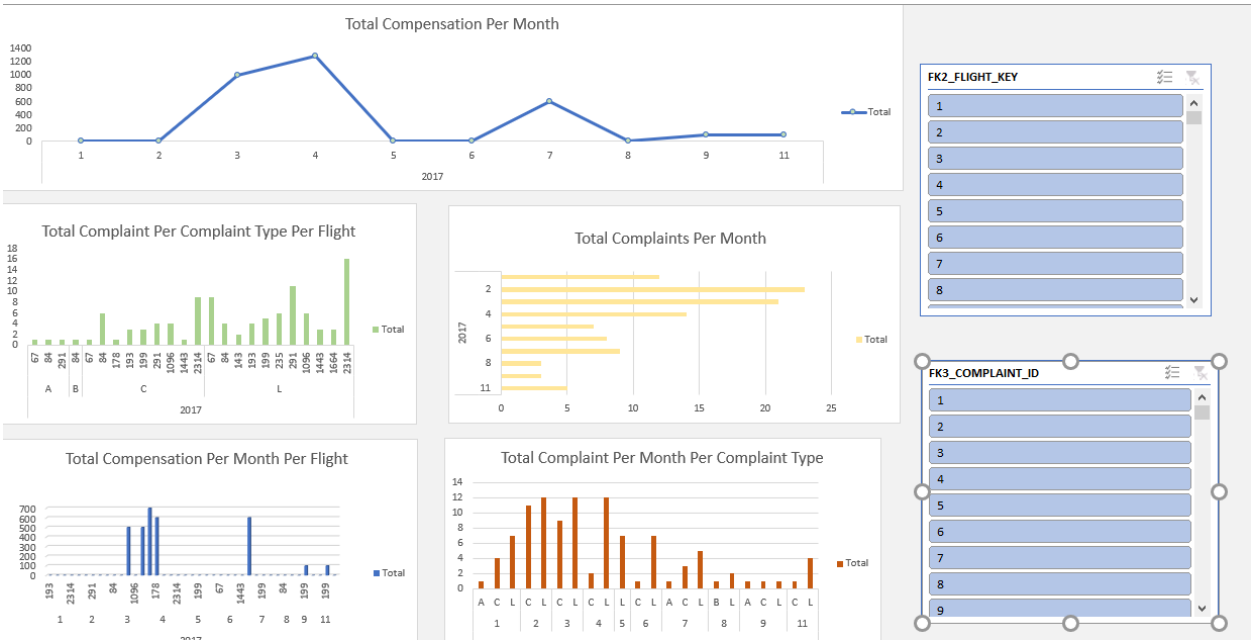
EDIT	FACT_ID	FK1_TIME_ID	FK2_FLIGHT_KEY	FK3_COMPLAINT_ID	NOOFCOMPLAINTS	TOTALCOMPENSATION
	1	15	1	1	1	0
	2	15	2	2	1	0
	3	15	3	3	1	0
	4	15	4	4	1	0
	5	15	5	5	1	0
	6	15	6	6	1	0
	7	15	7	7	1	0
	8	16	8	8	1	0
	9	16	9	9	1	0
	10	16	10	10	1	0
	11	16	11	11	1	0
	12	16	12	12	1	0

Task 4

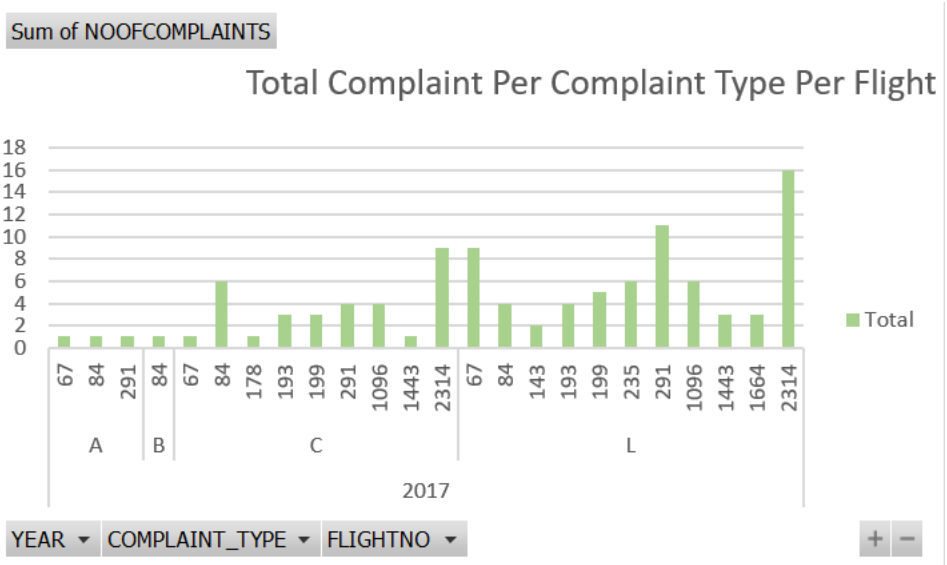
Data Analysis:

Dashboard:

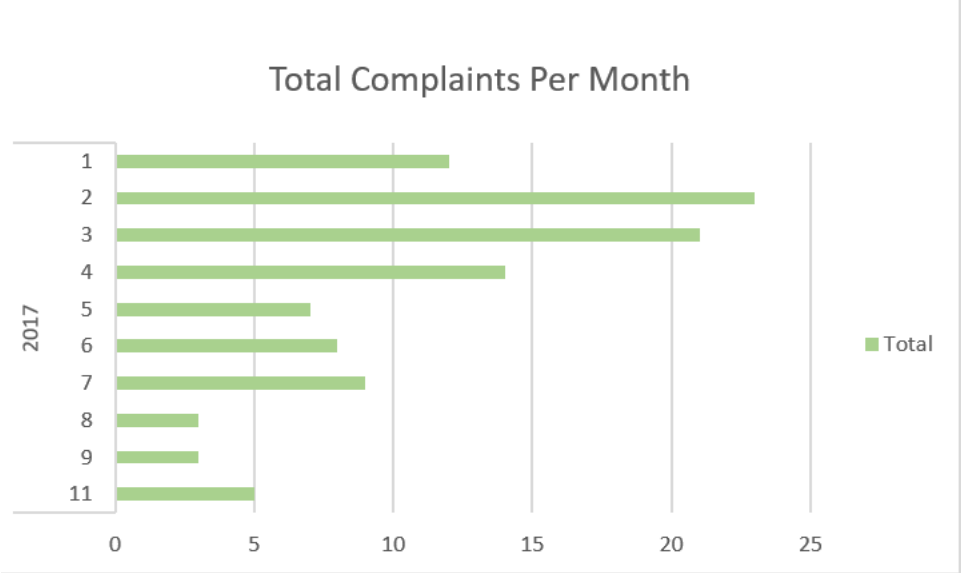
This dashboard facilitates the company to make multi-dimensional analysis based on flights, complaints and time.



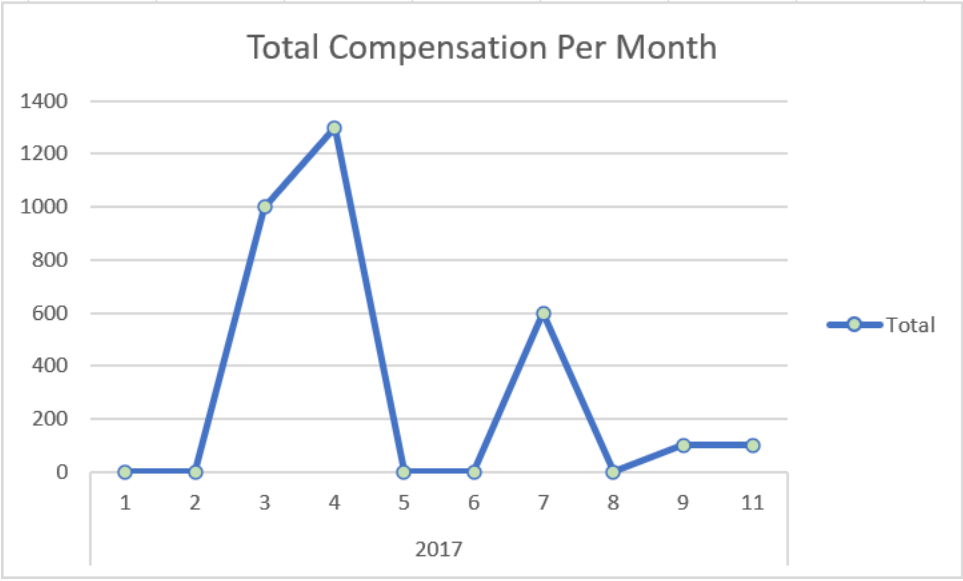
- Report 1



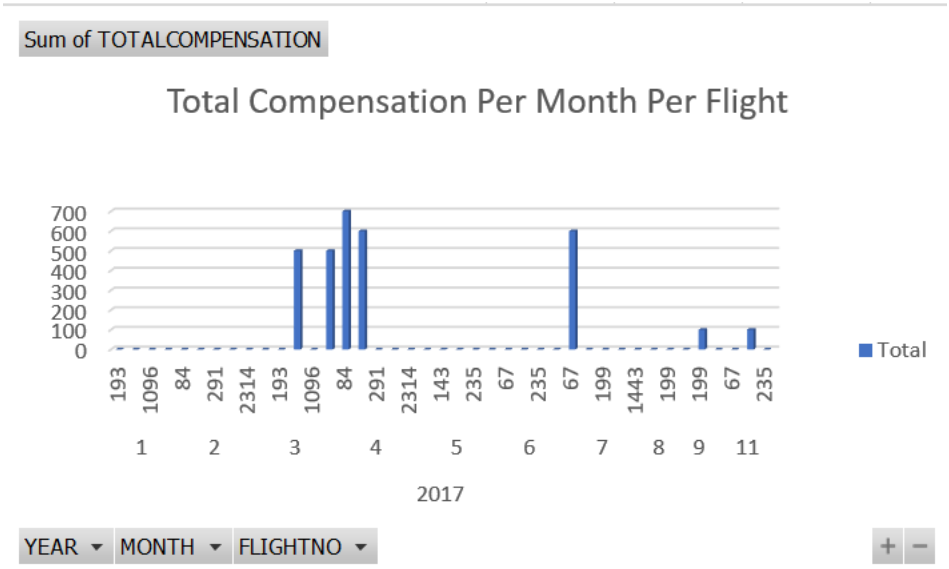
- Report 2



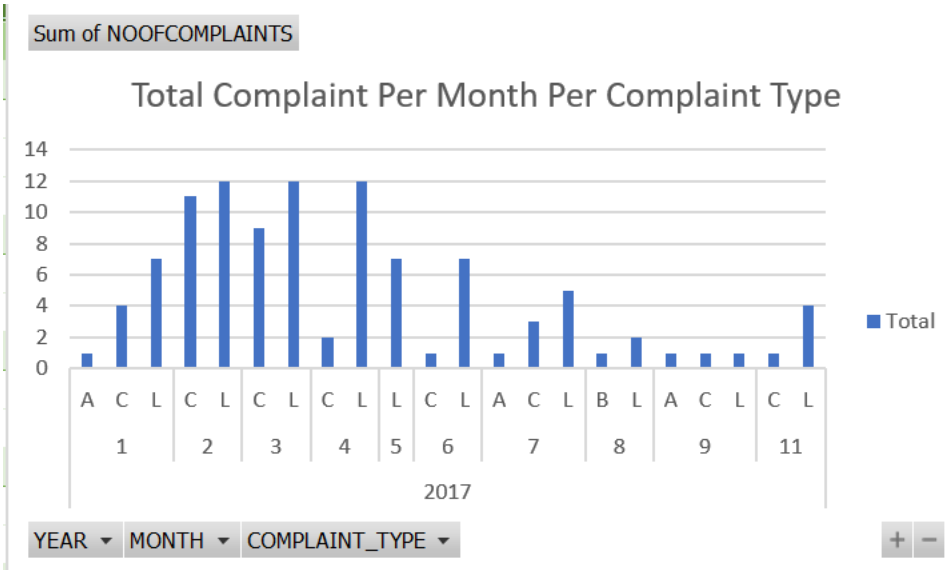
- Report 3



- Report 4



- Report 5



Task 5

Data Warehouse Approaches

In Data Warehouse, there are two major approaches when it comes to designing. They are Inmon Method and Kimball Method

With regards to effective corporate execution, it is urgent to decide the fitting methodology as per the necessity of the project. These assists cut with bringing down project cost just as save a ton of time. The two methods have their own points of interest and separating factors, so figuring out which strategy to utilize decides the fate of the company.

Bill Inmon's Method

The Bill Inmon's architecture, otherwise called the top-down design, arranges information utilizing ER modelling. In this architecture, a normalized data model is planned before the dimensional data marts where all the necessary data are made from the data warehouse. The strategy authorizes information distribution centre as a concentrated data warehouse where it catches the detailed information at the most reduced degree of detail henceforth, acquiring the name, detailed data warehouse. Thus, it gives a legitimate structure to conveying business insight all things considered at the focal point of the corporate information factory (CIF). Just, it is beginning with building a major, brought together endeavour data warehouse where all accessible information from transaction system is united into a subject-situated, incorporated, time-variation that helps decision making.

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.
- This approach requires experts to effectively manage a data warehouse.

Ralph Kimball Method

The Ralph Kimball architecture, otherwise called as bottom-up design of Data Warehouse, forms data marts first based on the business prerequisites. In this design, the key business questions and the key business measures are recognized before the essential information sources are assessed. Once, the information sources are breaking down and reported, the Extract, Transform and Load (ETL)software is used to bring information from numerous sources and load into a staging area. From that point onward, the data purification process happens where information is isolated into clean and error table. The information in error table is then recleaned and changed into the perfect information. From here, information is stacked into a dimensional which is not normalized. The dimensional modelling is finished utilizing the star schema. 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.

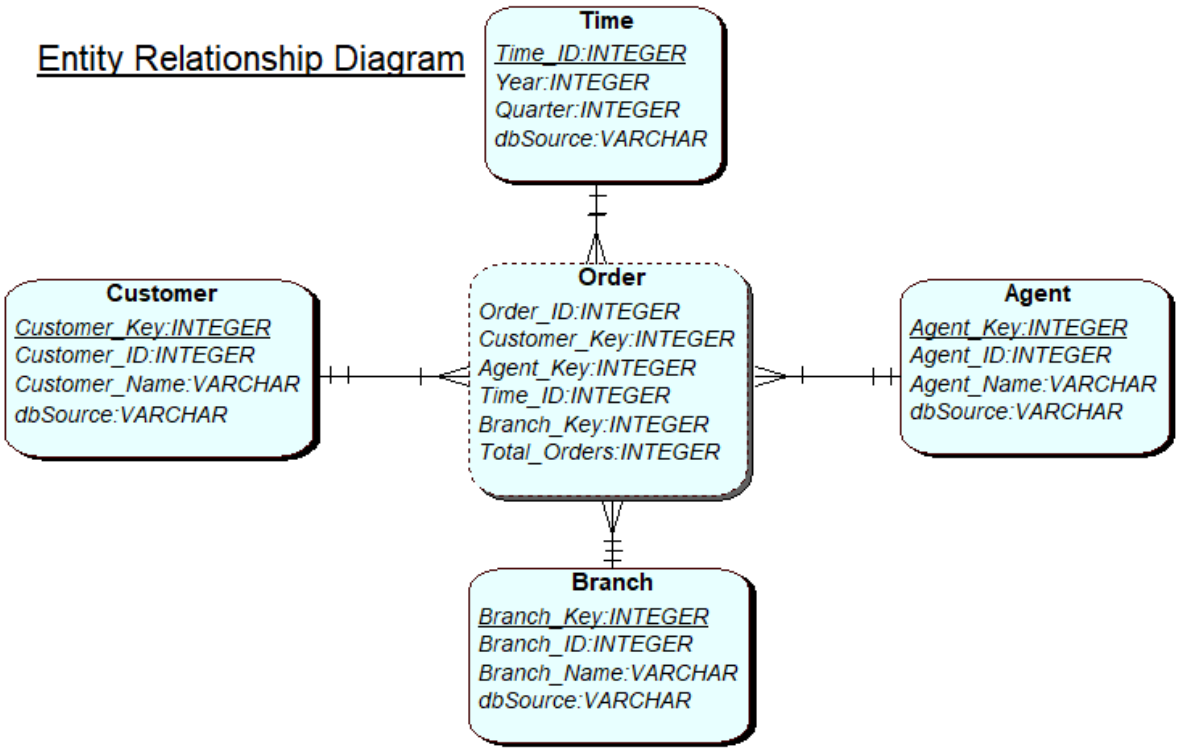
LBU Business:

Questions:

1. Design the star schema for the DW to be implemented....
2. Define, fact table, dimension(s), attributes, keys and measures
3. Number of orders made in UK first quarter of the year, in comparison with last year?
4. Who is our best customer, in first quarter of this year?
5. Total number of orders made in first quarter of the year, in comparison with last year for each branch?
6. Who is our best Agent, in the first quarter of this year?

Solutions:

Solution 1:



Solution 2:

Fact: Order Table (Order_ID, Customer_Key, Agent_Key, Time_ID, Branch_Key, Total_Orders)

Dimensions: Customer (Customer_Key, Customer_ID, Customer_Name, dbSource)

Agent (Agent_Key, Agent_ID, Agent_Name, dbSource)

Branch (Branch_Key, Branch_ID, Branch_Name, dbSource)

Time (Time_ID, Year, Quarter, dbSource)

Measures: Total_Orders

Solution 3:

SELECT t.Quarter, SUM (Total_Orders)

FROM Order o, Time t

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

Solution 4:

```
SELECT SUM (Total_Orders), Customer_Key
FROM
(
SELECT SUM (Total_Orders), Customer_Key
RANK OVER (ORDER BY SUM (Total_Orders) DESC) AS Rank
FROM Order o, Time t,
WHERE o.Time_ID = t.Time_ID
AND t.Quarter = “Q1”
AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year
)
WHERE Rank <=1;
```

Solution 5:

```
SELECT t.Quarter, SUM (Total_Orders), Branch_Key
FROM Order o, Time t
```

```
WHERE o.Time_ID = t.Time_ID
AND t.Quarter = “Q1”
AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year
GROUP BY  t.Quarter, Branch_Key
UNION
SELECT t.Quarter, SUM (Total_Orders), Branch_Key
FROM Order o, Time t
WHERE o.Time_ID = t.Time_ID
AND TO_CHAR(SYSDATE, ‘YYYY’) -1 = t.Year
GROUP BY  t.Quarter, Branch_Key;
```

Solution 6:

```
SELECT SUM (Total_Orders),Agent_Key
FROM
(
SELECT SUM (Total_Orders), Agent_Key
RANK OVER (ORDER BY SUM (Total_Orders) DESC) AS Rank
FROM Order o, Time t,
WHERE o.Time_ID = t.Time_ID
AND t.Quarter = “Q1”
AND TO_CHAR(SYSDATE, ‘YYYY’) = t.Year
)
WHERE Rank <=1;
```

Exercise: Data Warehouse design for a wholesale furniture company

Questions:

1. Identify facts, dimensions and measures
2. For each fact:
    a) Produce the attribute tree and fact schema
    b) Design the star or snowflake schema and write the following SQL queries:

        i. Find the quantity, the total income and discount with respect to each city, type of furniture and the month
        ii. Find the average quantity, income and discount with respect to each country, furniture material and year
        iii. Determine the 5 most sold furniture during the May month

Solutions:

Question 1:

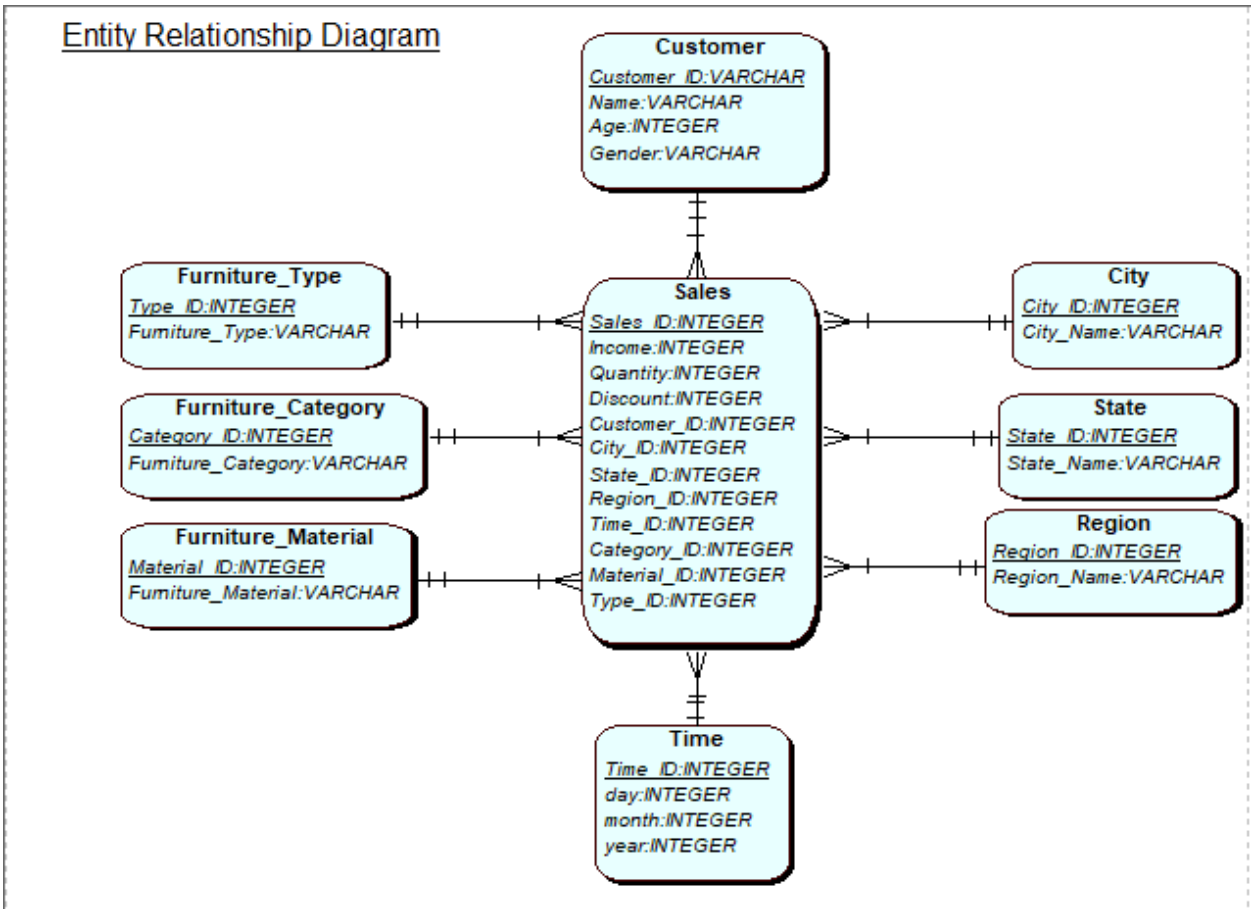
FACT Sales

MEASURES Quantity, Income, Discount

DIMENSIONS Furniture (Type, Category, Material) Customer (Age, Sex, City → Region → State) Time (Day → Month → Year)

Question 2 (a):

Question 2(b):



Question 2(b) i:

```
SELECT SUM(Quantity), SUM(Income), SUM(Discount), City_ID, Type_ID, Time_ID
FROM Sales
GROUP BY City_ID, Type_ID, Time_ID;
```

Question 2(b) ii:

```
SELECT AVG(Quantity), AVG(Income), AVG(Discount), State_ID, Material_ID, Time_ID
FROM Sales
GROUP BY  State_ID, Material_ID, Time_ID;
```

Question 2(b) iii:

```
SELECT Type_ID, SUM(Quantity) as Total
FROM (
SELECT Type_ID, 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;
```