

DATA WAREHOUSE PROJECT

-Ruchika Dhungana

Table of Contents

Project Introduction.....	2
Aims/Objectives.....	2
Data Warehouse.....	2
Main Objective:.....	2
OLAP:.....	3
OLTP:.....	3
OLAP Vs OLTP:.....	3
KPI 2: Ensure customer satisfaction.....	4
Task 1.....	4
REPORTS:.....	4
STAR SCHEMA:.....	5
DATA DICTIONARY:.....	5
REPORT:.....	9
Task 2.....	11
STAGING AREA SETUP:.....	11
STAR SCHEMA SETUP:.....	14
Task 3.....	15
EXTRACTION:.....	15
DATA PURIFICATION:.....	18
TRANSFORMATION:.....	25
LOAD:.....	28
Task 4.....	31
Data Analysis:.....	31
Dashboard:.....	33
Task 5.....	33
Data Warehouse Approaches.....	33
Assignment Portfolio.....	36

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

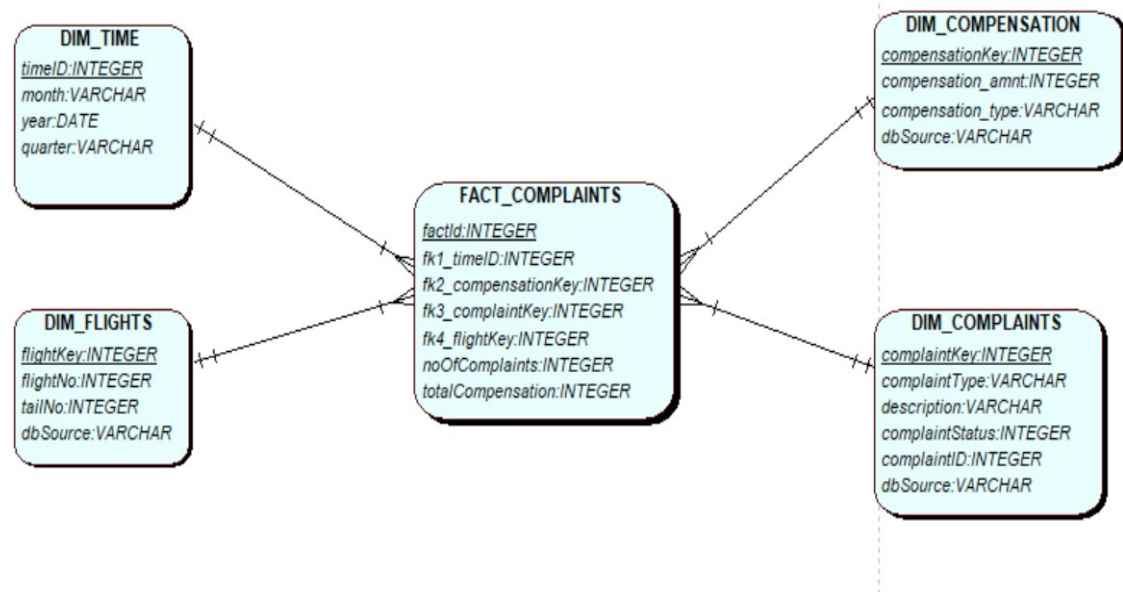
REPORTS:

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:

Entity Relationship Diagram



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	VARCHAR	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 irregular 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

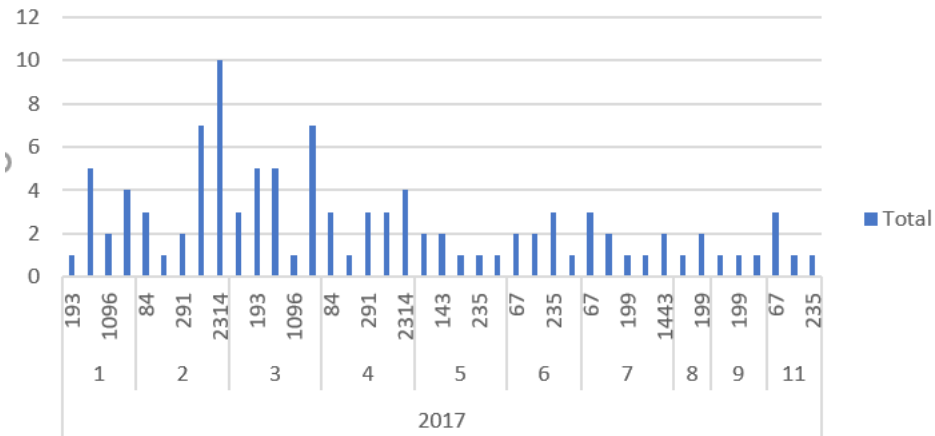
Star Schema Table	Attribute Name	Data Type	Key	DQ Source	Data Mapping	Data quality Issues	Transformation
FACT_COMPLAIN	FactId	INTEGER	Yes	Automatically generate	n/a	n/a	Create a sequence

TS				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
	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 holds all the data related to the customer complaints.						
Notes:							

REPORT:

- EXCEL REPORT

Total Complaint Per Flight Per Month



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

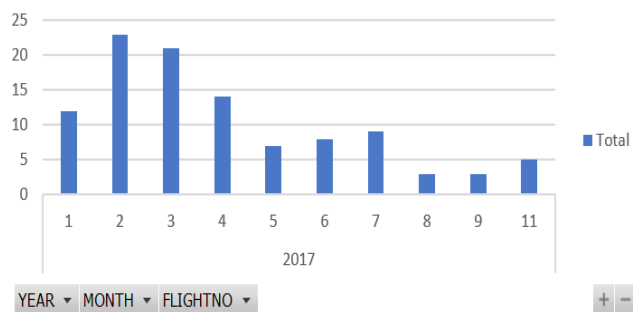
- Rollup Report

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

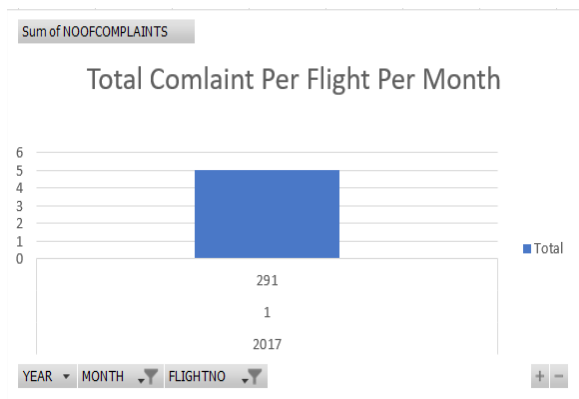
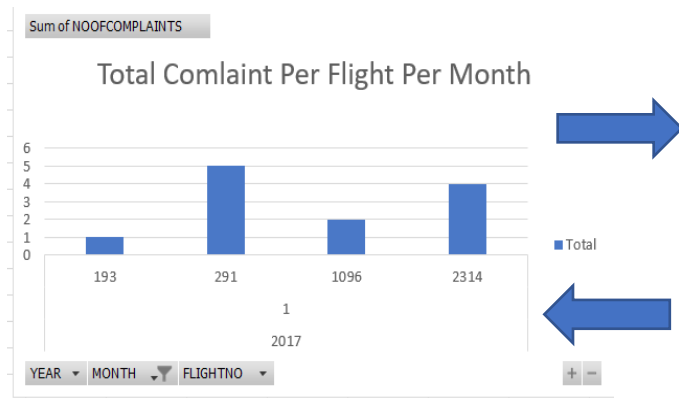


Sum of NOOFCOMPLAINTS

Total Complaint Per Flight Per Month



- 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

ORACLE APEX

App Builder

SQL Workshop

Team Development

App Gallery

CT202617

SQL ScriptsResults

Script: kpi2Status: Complete

View:

Detail

Summary

 Rows: 15

Go

Create App from ScriptEdit Script

Number	Elapsed	Statement	Feedback	Rows
1	0.01	DROP TABLE FlyU_flights CASCADE CONSTRAINTS	Table dropped.	0
2	0.02	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	0.00	CREATE TABLE FlyU_flights (flight_the_year NUMBER(4), t	Table created.	0
7	0.02	ALTER TABLE FlyU_flights ADD CONSTRAINT pk1_flights PRIMARY	Table altered.	0
8	0.00	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	0.00	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	0.01	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	0.00	INSERT INTO Customer VALUES (104, 'NY104', 'BUSINESS', 'HP',	1 row(s) inserted.	1
15	0.00	INSERT INTO Customer VALUES (105, 'NY105', 'BUSINESS', 'SkyB	1 row(s) inserted.	1

Download

row(s) 1 - 15 of 245Next

245

Statements Processed

245

Successful

0

With Errors

FlyU_flights

FLYU_FLIGHTS

TableDataIndexesModelConstraintsGrantsStatisticsUI DefaultsTriggersDependenciesSQLREST

Add ColumnModify ColumnRename ColumnDrop ColumnRenameCopyDropTruncateCreate 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

+ v

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	N3GVAA	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	N3YAA	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	N3GVAA	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	N3YVAA	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

+ v

Table

Data

Indexes

Model

Constraints

Grants

Statistics

UI Defaults

Triggers

Dependencies

SQL

REST

Add Column

Modify Column

Rename Column

Drop Column

Rename

Copy

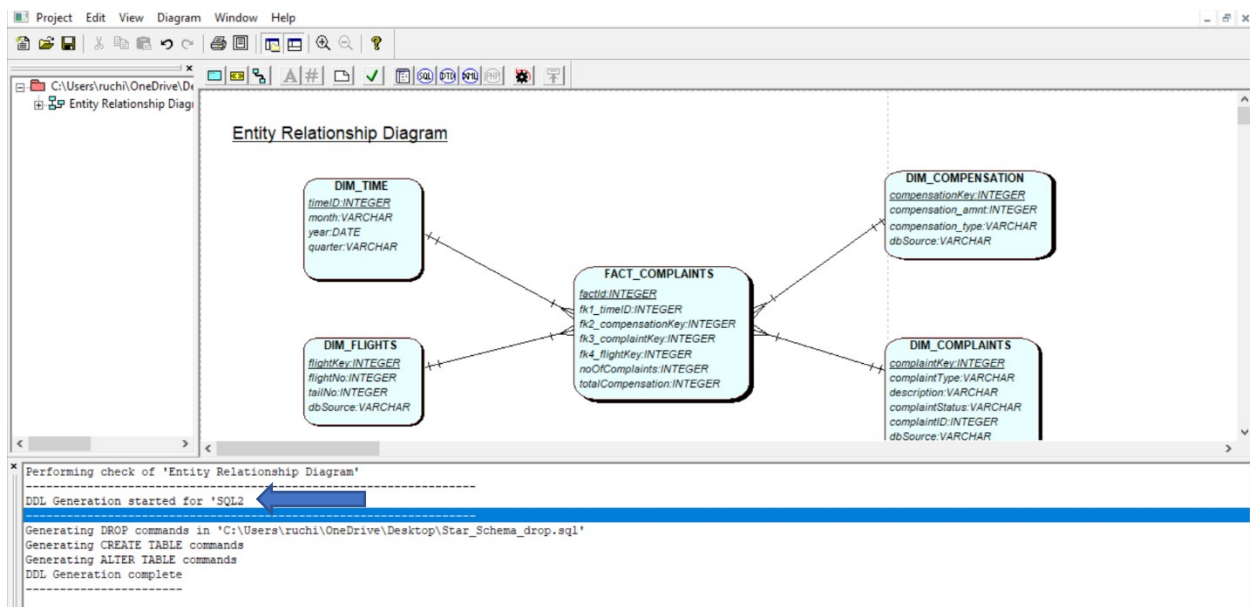
Drop

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](#)



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

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Scripts Results

Script: **Star_Schema** Status: **Complete**

View: ☐ Detail ☒ Summary Rows: 10

Number	Elapsed	Statement	Feedback	Rows
1	0.02	DROP TABLE DIM_TIME CASCADE CONSTRAINTS	Table dropped.	0
2	0.01	DROP TABLE DIM_COMPENSATION CASCADE CONSTRAINTS	Table dropped.	0
3	0.01	DROP TABLE DIM_COMPLAINTS CASCADE CONSTRAINTS	Table dropped.	0
4	0.01	DROP TABLE DIM_FLIGHTS CASCADE CONSTRAINTS	Table dropped.	0
5	0.02	DROP TABLE FACT_COMPLAINTS CASCADE CONSTRAINTS	Table dropped.	0
6	0.01	CREATE TABLE DIM_TIME(timeID: INTEGER NOT NULL, month: VARCH	Table created.	0
7	0.01	CREATE TABLE DIM_COMPENSATION(compensationKey: INTEGER NOT	Table created.	0
8	0.01	CREATE TABLE DIM_COMPLAINTS(complaintKey: INTEGER NOT NULL	Table created.	0
9	0.01	CREATE TABLE DIM_FLIGHTS(flightKey: INTEGER NOT NULL, flig	Table created.	0
10	0.02	CREATE TABLE FACT_COMPLAINTS(factId: INTEGER NOT NULL, noO	Table created.	0

Download

row(s) 1 - 10 of 14 [Next](#)

14	14	0
Statements Processed	Successful	With Errors

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

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.

ORACLE APEX

App BuilderSQL WorkshopTeam DevelopmentApp Gallery

SchemaC7202617

SQL Commands

Rows10Clear CommandFind TablesSaveRun

--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_amt, compensation_type, flight_Key, dbSource)
 VALUES(STAGE_COMPENSATION_SEQ.nextval, j.compensation_amt, j.compensation_type, nextval, 'flyU_flights_2020');

END LOOP;

END LOOP;

begin
STAGING_DATA_INSERT;
end;

ResultsExplainDescribeSaved SQLHistory

Procedure created.

0.04 seconds

There are three staging tables:

- **Stage Complaints**

ORACLE APEX

App BuilderSQL WorkshopTeam DevelopmentApp Gallery

SchemaC7202617

SQL Commands

Rows10Clear CommandFind TablesSaveRun

--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

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

c7202617

c7202617

en

Copyright © 1999, 2019, Oracle. All rights reserved.

Application Express 19.1.0.00.15

■ Stage Compensation

ORACLE APEX

App Builder

SQL Workshop

Team Development

App Gallery

Q

C7202617

SQL Commands

Schema C7202617

Rows 10

Clear Command

Find Tables

Save

Run

--DELETE STAGING COMPENSATION TABLE--
DROP TABLE STAGE_COMPENSATION CASCADE CONSTRAINTS;

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

--CREATE STAGING COMPENSATION TABLE--
CREATE TABLE STAGE_COMPENSATION
(
 compensationKey INTEGER NOT NULL,
 compensation_amt NUMBER(*,0),
 flight_key INTEGER NOT NULL,
 dbsource VARCHAR(17),

CONSTRAINTS pk_STAGE_COMPENSATION PRIMARY KEY (compensationKey)
)

ALTER TABLE STAGE_COMPENSATION ADD CONSTRAINT "FK1_STG_COMPENSATION_TO_FLIGHT" FOREIGN KEY (flight_key)
 REFERENCES STAGE_FLIGHTS(flightkey);

Results

Explain

Describe

Saved SQL

History

Table created.

0.01 seconds

C7202617 C7202617 en

Copyright © 1999, 2019, Oracle. All rights reserved.

Application Express 19.1.0.00.15

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

ORACLE APEX

App Builder

SQL Workshop

Team Development

App Gallery

Search

Schema

C7202617

SQL Commands

Rows: 10

Clear Command

Find Tables

Save

Run

```
--DELETE STAGING FLIGHTS TABLE--
DROP TABLE STAGE_FLIGHTS CASCADE CONSTRAINTS;

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

--CREATE STAGING FLIGHTS TABLE--
CREATE TABLE STAGE_FLIGHTS
(
  flightkey INTEGER NOT NULL PRIMARY KEY,
  flightno NUMBER(10,0),
  year NUMBER(4,0),
  month NUMBER(2,0),
  day NUMBER(2,0),
  tailno VARCHAR2(10),
  dbsource VARCHAR(17)
)
```

Results

Explain

Describe

Saved SQL

History

Table created.

0.02 seconds

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

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

The screenshot shows the Oracle APEX SQL Workshop interface. The 'SQL Commands' tab is active, displaying a SQL script to create two tables: `bad_complaints` and `good_complaints`. The script includes truncating existing tables, dropping constraints, and creating new tables with specific columns and data types. A blue arrow points to the 'Run' button at the bottom right of the SQL editor.

```
--CREATING BAD DATA TABLE FOR COMPLAINTS TABLE--
truncate table bad_complaints;
DROP table bad_complaints CASCADE CONSTRAINTS;
create table bad_complaints
(
  "BAD_ID" NUMBER(*,0) NOT NULL PRIMARY KEY,
  "COMPLAINTKEY" NUMBER(*,0),
  "COMPLAINTID" NUMBER(*,0) NOT NULL ENABLE,
  "COMPLAINTTYPE" VARCHAR2(5),
  "DESCRIPTION" VARCHAR2(200),
  "COMPLAINTSTATUS" VARCHAR2(7),
  "FLIGHT_KEY" INTEGER NOT NULL,
  "DESOURCE" VARCHAR2(17),
  "ERROR_DESCRIPTION" VARCHAR2(15),
  "STATUS" VARCHAR2(10),
  "RESOLUTION_DATE" DATE
);

--CREATING GOOD DATA TABLE FOR COMPLAINTS TABLE--
truncate table good_complaints;
DROP table good_complaints CASCADE CONSTRAINTS;
create table good_complaints
(
  COMPLAINTKEY NUMBER NOT NULL PRIMARY KEY,
  COMPLAINTID NUMBER(*,0) NOT NULL ENABLE,
  COMPLAINTTYPE VARCHAR2(5),
  DESCRIPTION VARCHAR2(200),
  COMPLAINTSTATUS VARCHAR2(7),
  FLIGHT_KEY INTEGER NOT NULL,
  DESOURCE VARCHAR2(17)
);
```

- Bad/Good Data Identification

The screenshot shows the Oracle APEX SQL Workshop interface with the 'Object Browser' tab active. It displays a procedure named `COMPLAINTS_DATA_QUALITY_CHECK`. The procedure's code is shown, which includes logic to identify bad and good data based on specific criteria and insert them into the respective tables.

```
1 create or replace procedure complaints_data_quality_check (pv_rows OUT NUMBER)
2 is
3 BEGIN
4   pv_rows := 0;
5
6   INSERT INTO bad_complaints
7   (SELECT bad_complaints_seq.nextval, stage_complaints.complaintkey, stage_complaints.complaintid, stage_complaints.complainttype, stage_complaints.description, stage_complaints.complaintstatus, stage_complaints.flight_key, stage_complaints.desource
8   FROM stage_complaints
9   WHERE complainttype IS NULL
10  OR description IS NULL
11  OR complaintstatus IS NULL);
12   pv_rows := TO_CHAR(SQL%rowcount);
13
14   INSERT INTO good_complaints
15   (SELECT bad_complaints_seq.nextval, stage_complaints.complaintkey, stage_complaints.complaintid, stage_complaints.complainttype, stage_complaints.description, stage_complaints.complaintstatus, stage_complaints.flight_key, stage_complaints.desource
16   FROM stage_complaints
17   WHERE description = 'cancelled' AND complainttype <> 'C');
18   pv_rows := pv_rows + TO_CHAR(SQL%rowcount);
19
20   DELETE FROM good_complaints;
21   INSERT INTO good_complaints
22   (
23   SELECT * FROM stage_complaints
24   WHERE complainttype IS NOT NULL
25   AND description IS NOT NULL
26   AND complaintstatus IS NOT NULL
27   AND description = 'cancelled' AND complainttype = 'C'
28   );
29 END;
```

ORACLE APEX SQL Workshop Team Development App Gallery

SQL Commands

Rows: 10 Clear Command Find Tables

```
--EXECUTE PROCEDURE--
DECLARE
  noOfRows NUMBER(5,2);
BEGIN
  complaints_data_quality_check (noOfRows);
  DBMS_OUTPUT.PUT_LINE('No of bad data: '||noOfRows);
END;
```

Results Explain Describe Saved SQL History

No of bad data: 183
Statement processed.
0.03 seconds

BAD TABLE

BAD_COMPLAINTS

Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
Query	Count Rows	Insert Row									
EDIT	BAD_ID	COMPLAINTKEY	COMPLAINTID	COMPLAINTTYPE	DESCRIPTION	COMPLAINTSTATUS	FLIGHT_KEY	DBSOURCE	ERROR_DESCRIPTION	STATUS	RESOLUTION_DATE
	1	1	1	-	late	open	1	flyU_flights_2020	null values	not fixed	12/19/2020
	2	2	2	-	late	open	2	flyU_flights_2020	null values	not fixed	12/19/2020
	3	3	3	-	late	open	3	flyU_flights_2020	null values	not fixed	12/19/2020
	4	4	4	-	late	open	4	flyU_flights_2020	null values	not fixed	12/19/2020
	5	5	5	-	late	open	5	flyU_flights_2020	null values	not fixed	12/19/2020
	6	6	6	-	late	open	6	flyU_flights_2020	null values	not fixed	12/19/2020
	7	7	7	-	late	open	7	flyU_flights_2020	null values	not fixed	12/19/2020
	8	8	8	-	late	open	8	flyU_flights_2020	null values	not fixed	12/19/2020
	9	9	9	-	late	open	9	flyU_flights_2020	null values	not fixed	12/19/2020
	10	10	10	-	late	open	10	flyU_flights_2020	null values	not fixed	12/19/2020
	11	11	11	-	late	open	11	flyU_flights_2020	null values	not fixed	12/19/2020
	12	12	12	-	late	open	12	flyU_flights_2020	null values	not fixed	12/19/2020

Reclean Bad Data

ORACLE APEX SQL Workshop Team Development App Gallery

SQL Commands

Rows: 100 Clear Command Find Tables

```
--Reclean bad data in Complaint bad table--
create or replace procedure reclean_complaint_bad_data(pv_rows OUT NUMBER)
is
BEGIN
  pv_rows := 0;
  update bad_complaints set DESCRIPTION = 'unknown', STATUS = 'fixed', RESOLUTION_DATE = sysdate where DESCRIPTION is null;
  pv_rows := TO_CHAR(SQL%RowCount);
  update bad_complaints set COMPLAINTTYPE='L', STATUS = 'fixed', RESOLUTION_DATE = sysdate where COMPLAINTTYPE is null and DESCRIPTION = 'late';
  pv_rows := pv_rows + TO_CHAR(SQL%RowCount);
  update bad_complaints set COMPLAINTTYPE='C', STATUS = 'fixed', RESOLUTION_DATE = sysdate where (COMPLAINTTYPE='A' OR COMPLAINTTYPE='B') and DESCRIPTION = 'cancelled';
  pv_rows := pv_rows + TO_CHAR(SQL%RowCount);
end;
```

Results Explain Describe Saved SQL History

Procedure created.
0.01 seconds

Transfer Recleaned Data

ORACLE APEX SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--Transferring recleaned Complaint bad data to good table
create or replace procedure RECLEANED_COMPLAINT_DATA_HERGE
as
begin
  HERGE INTO good_complaints c
  USING bad_complaints b
  ON (C.COMPLAINTKEY = B.COMPLAINTKEY)
  WHEN MATCHED THEN
    UPDATE SET
      C.COMPLAINTID = B.COMPLAINTID,
      C.COMPLAINTTYPE = B.COMPLAINTTYPE,
      C.DESCRPTION=B.DESCRPTION,
      C.COMPLAINTSTATUS=B.COMPLAINTSTATUS,
      C.FLIGHT_KEY = B.FLIGHT_KEY,
      C.DBSOURCE=B.DBSOURCE
  WHEN NOT MATCHED THEN
    INSERT VALUES
      (B.COMPLAINTKEY, B.COMPLAINTID, B.COMPLAINTTYPE, B.DESCRPTION, B.COMPLAINTSTATUS, B.FLIGHT_KEY, B.DBSOURCE);
  END;
begin
  RECLEANED_COMPLAINT_DATA_HERGE;
end;
```

Results Explain Describe Saved SQL History

Procedure created.

- Good Table

ORACLE APEX SQL Workshop Team Development App Gallery

Object Browser Schema C7202617

GOOD_COMPLAINTS							
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

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--CREATING BAD DATA TABLE FOR COMPLAINTS TABLE--
select * from bad_compensation;
DROP TABLE bad_compensation CASCADE CONSTRAINTS;
CREATE TABLE "BAD_COMPENSATION"
(
  "BAD_ID" NUMBER NOT NULL PRIMARY KEY,
  "COMPENSATIONKEY" NUMBER(*,0) NOT NULL ENABLE,
  "COMPENSATION_AMNT" VARCHAR2(20),
  "COMPENSATION_TYPE" VARCHAR2(20),
  "FLIGHT_KEY" INTEGER NOT NULL,
  "DBSOURCE" VARCHAR2(17),
  "ERROR_DESCRIPTION" VARCHAR2(15),
  "STATUS" VARCHAR2(10),
  "RESOLUTION_DATE" DATE
);

--CREATING BAD DATA TABLE FOR COMPLAINTS TABLE--
select * from good_compensation;
DROP TABLE good_compensation CASCADE CONSTRAINTS;
CREATE TABLE "GOOD_COMPENSATION"
(
  "COMPENSATIONKEY" NUMBER NOT NULL PRIMARY KEY,
  "COMPENSATION_AMNT" VARCHAR2(20),
  "COMPENSATION_TYPE" VARCHAR2(20),
  "FLIGHT_KEY" INTEGER NOT NULL,
  "DBSOURCE" VARCHAR2(17)
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

c7202617 c7202617 en Copyright © 1999, 2019, Oracle. All rights reserved. Application Express 19.10.00.15

■ Bad/Good Data Identification

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--CREATING PROCEDURE FOR BAD AND GOOD COMPENSATION TABLE DATA QUALIFICATION--
create or replace procedure compensation_data_quality_check (pv_rows OUT NUMBER)
IS
BEGIN
  pv_rows := 0;
  INSERT INTO bad_compensation
  (SELECT bad_compensation_seq.nextval, stage_compensation.compensationkey, stage_compensation.compensation_amnt, stage_compensation.compensation_type, stage_compensation.flight_key, stage_compensation.dbsource, 'null values', 'not fixed',
  sysdate
  FROM stage_compensation
  WHERE compensation_amnt IS NULL
  OR compensation_type IS NULL);
  pv_rows := TO_CHAR(SQL%ROWCOUNT);

  INSERT INTO bad_compensation
  (SELECT bad_compensation_seq.nextval, stage_compensation.compensationkey, stage_compensation.compensation_amnt, stage_compensation.compensation_type, stage_compensation.flight_key, stage_compensation.dbsource, 'inconsistent', 'not fixed',
  sysdate
  FROM stage_compensation
  WHERE REGEXP_LIKE (compensation_type, '^rb(ook)?$'));
  pv_rows := pv_rows + TO_CHAR(SQL%ROWCOUNT);

  DELETE FROM good_compensation;
  INSERT INTO good_compensation
  (SELECT * FROM stage_compensation
  WHERE compensation_amnt IS NOT NULL
  AND compensation_type IS NOT NULL
  AND NOT REGEXP_LIKE (compensation_type, '^rb(ook)?$'));
END;

--EXECUTE PROCEDURE--
DECLARE
  noOfRows NUMBER(5,2);
BEGIN
  compensation_data_quality_check (noOfRows);
  DBMS_OUTPUT.PUT_LINE('No of bad data: '||noOfRows);
END;
```

Results Explain Describe Saved SQL History

No of bad data: 8

Statement processed.

■ BAD TABLE

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

Object Browser Schema C7202617

BAD_COMPENSATION

Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
Query	Count Rows	Insert Row									
EDIT	BAD_ID	COMPENSATIONKEY	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE	ERROR_DESCRIPTION	STATUS	RESOLUTION_DATE		
	1	94	0	-	109	flyU_flights_2020	null values	not fixed	12/19/2020		
	2	100	0	-	115	flyU_flights_2020	null values	not fixed	12/19/2020		
	3	101	0	-	116	flyU_flights_2020	null values	not fixed	12/19/2020		
	4	102	0	-	117	flyU_flights_2020	null values	not fixed	12/19/2020		
	5	103	0	-	118	flyU_flights_2020	null values	not fixed	12/19/2020		
	6	87	0	rebook	102	flyU_flights_2020	inconsistent	not fixed	12/19/2020		
	7	88	0	rebook	103	flyU_flights_2020	inconsistent	not fixed	12/19/2020		
	8	89	0	rebook	104	flyU_flights_2020	inconsistent	not fixed	12/19/2020		

Download

Reclean Bad Data

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 100 Clear Command Find Tables Save Run

```
--Reclean bad data in Compensation bad table--
create or replace procedure reclean_compensation_bad_data(pv_rows OUT NUMBER)
is
    pv_rows := 0;
begin
    update bad_compensation set COMPENSATION_TYPE = 'unknown', STATUS = 'fixed', RESOLUTION_DATE = sysdate where COMPENSATION_TYPE is null;
    pv_rows := TO_CHAR(SQL%ROWCOUNT);

    update bad_compensation set COMPENSATION_TYPE = 'rebooked', STATUS = 'fixed', RESOLUTION_DATE = sysdate where REGEXP_LIKE (COMPENSATION_TYPE, '^reb(ook)?$');
    pv_rows := pv_rows + TO_CHAR(SQL%ROWCOUNT);

end;
```

```
--EXECUTE PROCEDURE--
set serveroutput on;
declare
    pv_rows NUMBER(9,3);
begin
    reclean_compensation_bad_data(pv_rows);
    dbms_output.put_line('No. of Bad data cleaned: ' || pv_rows);
end;
```

Results Explain Describe Saved SQL History

No. of bad data cleaned: 8
Statement processed.

Transfer Recleaned Data

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--Transferring recleaned bad Compensation data to good table
create or replace procedure RECLEANED_COMPENSATION_DATA_MERGE
as
begin
    MERGE INTO good_compensation C
    USING bad_compensation B
    ON (C.COMPENSATIONKEY = B.COMPENSATIONKEY)
    WHEN MATCHED THEN
        UPDATE SET
            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.COMPENSATIONKEY, B.COMPENSATION_AMNT, B.COMPENSATION_TYPE, B.FLIGHT_KEY, B.DBSOURCE);
end;
```

Results Explain Describe Saved SQL History

Procedure created.

0.07 seconds

- Good Table

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'ORACLE APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'App Gallery'. The 'Object Browser' on the left shows the 'GOOD_COMPENSATION' table under the 'C7202617' schema. The table is displayed in the 'Data' tab, showing columns: EDIT, COMPENSATIONKEY, COMPENSATION_AMNT, COMPENSATION_TYPE, FLIGHT_KEY, and DBSOURCE. The table contains 12 rows of data.

EDIT	COMPENSATIONKEY	COMPENSATION_AMNT	COMPENSATION_TYPE	FLIGHT_KEY	DBSOURCE
	95	600	refund	110	flyU_flights_2020
	96	700	refund	111	flyU_flights_2020
	97	0	rebooked	112	flyU_flights_2020
	98	0	rebooked	113	flyU_flights_2020
	99	600	refund	114	flyU_flights_2020
	104	100	voucher	119	flyU_flights_2020
	105	100	voucher	120	flyU_flights_2020
	88	0	rebooked	103	flyU_flights_2020
	101	0	unknown	116	flyU_flights_2020
	103	0	unknown	118	flyU_flights_2020
	87	0	rebooked	102	flyU_flights_2020
	102	0	unknown	117	flyU_flights_2020

FLIGHTS

- Table Creation

The screenshot shows the Oracle APEX SQL Workshop interface with the 'SQL Commands' tab selected. The 'Schema' is set to 'C7202617'. The SQL command area contains the following code:

```
--CREATING BAD DATA TABLE FOR FLIGHTS TABLE--
truncate table bad_flights;
DROP table bad_flights CASCADE CONSTRAINTS;
create table bad_flights
(
  "BAD_ID" NUMBER(*,0) NOT NULL PRIMARY KEY,
  "FLIGHTKEY" 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--
truncate table good_flights;
DROP table good_flights CASCADE CONSTRAINTS;
create table good_flights
(
  FLIGHTKEY 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)
);
```

The 'Results' tab at the bottom shows the message: 'Table created.'

- Bad/Good Data Identification

ORACLE APEX SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--CREATING PROCEDURE FOR BAD AND GOOD FLIGHTS TABLE DATA QUALIFICATION--
create or replace procedure flights_data_quality_check (pv_rows OUT NUMBER)
IS
BEGIN
  pv_rows := 0;
  INSERT INTO bad_flights
  (
    SELECT BAD_FLIGHTS_SEQ.nextval, STAGE_FLIGHTS.FLIGHTKEY, STAGE_FLIGHTS.FLIGHTNO, STAGE_FLIGHTS.YEAR, STAGE_FLIGHTS.MONTH, STAGE_FLIGHTS.DAY, STAGE_FLIGHTS.TAILNO, STAGE_FLIGHTS.DBSOURCE, 'null values', 'not fixed', sysdate
    FROM STAGE_FLIGHTS
    WHERE FLIGHTNO IS NULL
    OR YEAR IS NULL
    OR MONTH IS NULL
    OR DAY IS NULL
    OR TAILNO IS NULL;
  )
  DELETE FROM good_flights;
  INSERT INTO good_flights
  (
    SELECT * FROM STAGE_FLIGHTS
    WHERE FLIGHTNO IS NOT NULL
    OR YEAR IS NOT NULL
    OR MONTH IS NOT NULL
    OR DAY IS NOT NULL
    OR TAILNO IS NOT NULL;
  )
  END;
```

Results Explain Describe Saved SQL History

Procedure created.

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

■ BAD TABLE

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

Object Browser Schema C7202617

BAD_FLIGHTS +

Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
Query Count Rows Insert Row											
Data											
This table contains no data.											

■ 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

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
--TRANSFORM_COMPLAINT--
DROP TABLE TRANSFORM_COMPLAINT CASCADE CONSTRAINTS;
CREATE TABLE TRANSFORM_COMPLAINT
(
  COMPLAINT_KEY NUMBER NOT NULL PRIMARY KEY,
  COMPLAINT_ID NUMBER NOT NULL,
  "COMPLAINTTYPE" VARCHAR2(5),
  "DESCRIPTION" VARCHAR2(200),
  "COMPLAINTSTATUS" VARCHAR2(7),
  "FLIGHT_KEY" INTEGER NOT NULL,
  "DBSOURCE" VARCHAR2(17)
);
```

Results Explain Describe Saved SQL History

Table created.

0.02 seconds

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```

--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 GOOD_COMPLAINTS);
END;

begin
TRANSFORM_COMPLAINT_TABLE;
end;

```

Results Explain Describe Saved SQL History

Procedure created.

0.01 seconds

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

c7202617 c7202617 en Copyright © 1999, 2019, Oracle. All rights reserved. Application Express 19.1.0.0.15

■ Transform Compensation

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```

--TRANSFORM_COMPENSATION--
DROP TABLE TRANSFORM_COMPENSATION CASCADE CONSTRAINTS;
CREATE TABLE TRANSFORM_COMPENSATION
(
"COMPENSATIONKEY" NUMBER(*,0) NOT NULL PRIMARY KEY,
"COMPENSATION_AMOUNT" VARCHAR2(20),
"COMPENSATION_TYPE" VARCHAR2(20),
"FLIGHT_KEY" INTEGER NOT NULL,
"DBSOURCE" VARCHAR2(17)
);

```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

```
--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
 Copyright © 1999, 2019, Oracle. All rights reserved.
 Application Express 19.1.0.00.15

- Transform Flights

ORACLE APEX

App Builder

SQL Workshop

Team Development

App Gallery

🔍

👤

🕒

C7202617

SQL Commands

Schema: C7202617

Rows: 10

Clear Command

Find Tables

Save

Run

```
--TRANSFORM_FLIGHTS--
DROP TABLE TRANSFORM_FLIGHTS CASCADE CONSTRAINTS;
TRUNCATE TABLE TRANSFORM_FLIGHTS;
CREATE TABLE TRANSFORM_FLIGHTS
(
    FLIGHTKEY 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)
);
```

Results

Explain

Describe

Saved SQL

History

Table created.

ORACLE APEX

App Builder

SQL Workshop

Team Development

App Gallery

🔍

👤

🕒

C7202617

SQL Commands

Schema: C7202617

Rows: 10

Clear Command

Find Tables

Save

Run

```
--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 GOOD_FLIGHTS);
END;

begin
TRANSFORM_FLIGHTS_TABLE;
end;
```

Results

Explain

Describe

Saved SQL

History

Procedure created.

🔍

📄

🌐

en

Copyright © 1999, 2019, Oracle. All rights reserved.

Application Express 19.1.0.00.15

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.

ORACLE APEX SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```

CREATE OR REPLACE PROCEDURE DIM_TABLE_INSERT
as
    v_count NUMBER;
BEGIN
    FOR i IN (SELECT * FROM transform_flights) LOOP
        INSERT INTO dim_flights(flightkey, flightno, tailno, dbsource) VALUES(i.flightkey, 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(timeid, year, month) VALUES(DIM_TIME_SEQ.nextval, i.year, i.month);
        end if;
        FOR j IN (SELECT * FROM transform_complaint WHERE flight_key=i.flightkey) LOOP
            INSERT INTO dim_complaints(complaintkey, complaintid, complainttype, description, complaintstatus, dbsource)
            VALUES(j.complaint_key, j.complaint_id, j.complainttype, j.description, j.complaintstatus, j.dbsource);
        END LOOP;

        FOR k IN (SELECT * FROM transform_compensation WHERE flight_key=i.flightkey) LOOP
            INSERT INTO dim_compensation (compensationkey, compensation_amt, compensation_type, dbsource) VALUES(k.compensationkey, k.compensation_amt, k.compensation_type, k.dbsource);
        END LOOP;
    END LOOP;
END;

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

```

Results Explain Describe Saved SQL History

1 row(s) inserted.

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

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

■ DIM_TIME

DIM_TIME			
Table	Data	Indexes	Model
Query	Count Rows	Insert Row	
Data			
EDIT	TIMEID	MONTH	YEAR
	121	1	2017
	122	2	2017
	123	3	2017
	124	4	2017
	125	5	2017
	126	6	2017
	127	7	2017
	128	8	2017
	129	9	2017
	130	11	2017
	131	12	2017
	132	10	2017

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

■ DIM_FLIGHTS

■ DIM_COMPLAINTS

<

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

■ DIM_COMPENSATION

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

FACT_COMPLAINTS

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

SQL Commands Schema C7202617

Rows 10 Clear Command Find Tables Save Run

```
CREATE SEQUENCE "FACT_COMPLAINT_SEQ" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 1;

CREATE OR REPLACE PROCEDURE FACT_TABLE_INSERT
as
BEGIN
  MERGE INTO FACT_COMPLAINTS fc
  USING (SELECT DISTINCT dt.timeid, cc.compensationkey, tc.complaint_key, tf.flightkey, COUNT(tc.complaint_key) AS NOOFCOMPLAINTS, SUM(cc.compensation_amt) as TOTALCOMPENSATION
  FROM transform_flights tf
  INNER JOIN transform_compensation cc ON(tf.flightkey=cc.flight_key)
  INNER JOIN transform_complaint tc ON(tf.flightkey=tc.flight_key)
  INNER JOIN dim_time dt
  ON(tf_year=dt.year AND tf_month=dt.month)
  GROUP BY dt.timeid, tf.flightkey, tc.complaint_key, cc.compensationkey) d
  ON(fc.fk1_timeid=d.timeid AND fc.fk2_compensationkey=d.compensationkey AND fc.fk3_complaintkey=d.complaint_key AND fc.fk4_flightkey=d.flightkey)
  WHEN MATCHED THEN
    UPDATE SET
      fc.noofofcomplaints=d.noofofcomplaints,
      fc.totalcompensation = d.totalcompensation
  WHEN NOT MATCHED THEN
    INSERT
    VALUES (FACT_COMPLAINT_SEQ.NEXTVAL, d.timeid, d.compensationkey, d.complaint_key, d.flightkey, d.noofofcomplaints, d.totalcompensation);
END;
truncate table fact_complaints;
BEGIN
  FACT_TABLE_INSERT;
END;
```

Results Explain Describe Saved SQL History

Statement processed.

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

ORACLE APEX App Builder SQL Workshop Team Development App Gallery

Object Browser Schema C7202617

FACT_COMPLAINTS

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

Query Count Rows Insert Row

Data

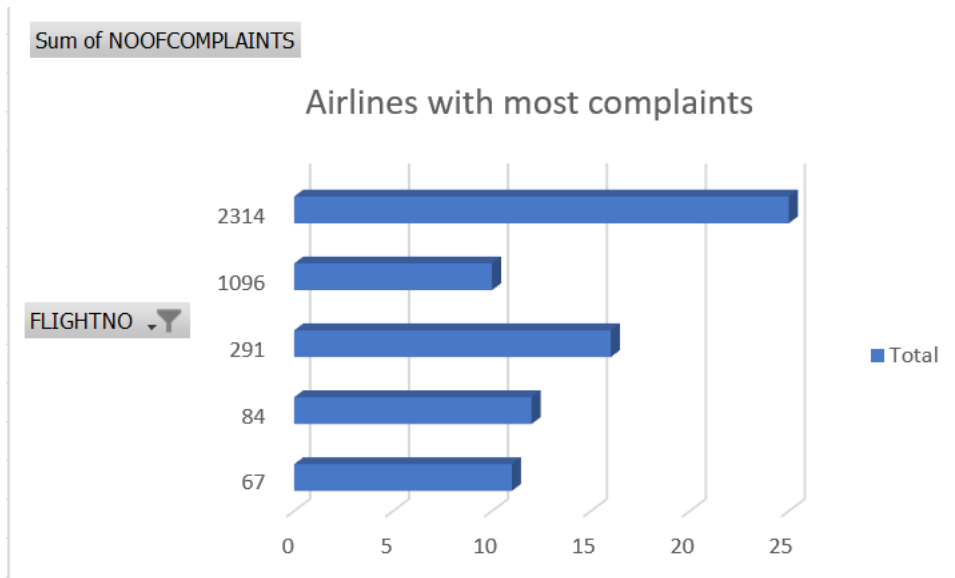
EDIT	FACTID	FK1_TIMEID	FK2_COMPENSATIONKEY	FK3_COMPLAINTKEY	FK4_FLIGHTKEY	NOOFCOMPLAINTS	TOTALCOMPENSATION
	106	135	15	15	15	1	0
	107	135	17	17	17	1	0
	108	136	30	30	30	1	0
	109	137	34	34	34	1	0
	110	137	37	37	37	1	0
	111	139	55	55	55	1	0
	112	134	7	7	7	1	0
	113	137	36	36	36	1	0
	114	137	39	39	39	1	0
	115	138	44	44	44	1	0
	116	142	64	64	64	1	0
	117	134	1	1	1	1	0

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

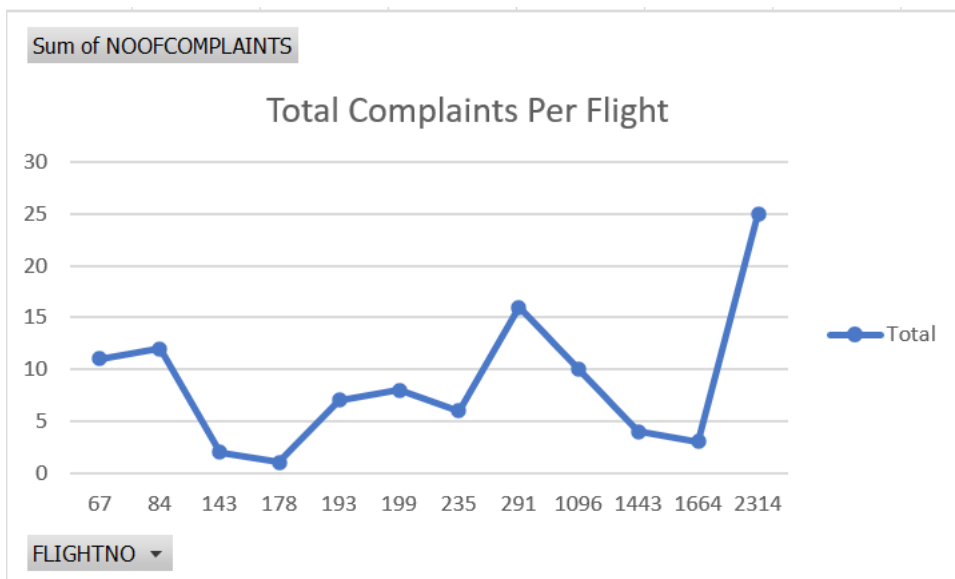
Task 4

Data Analysis:

- Report 1

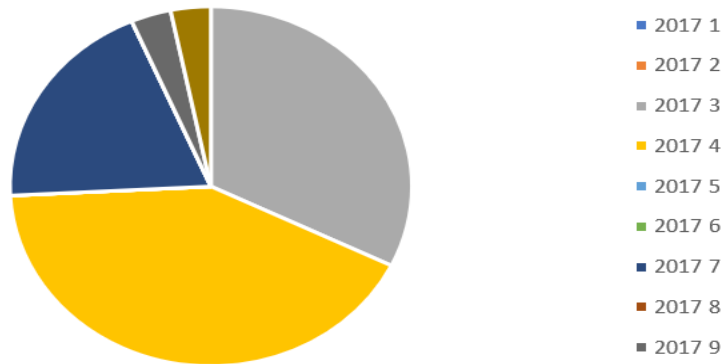


- Report 2



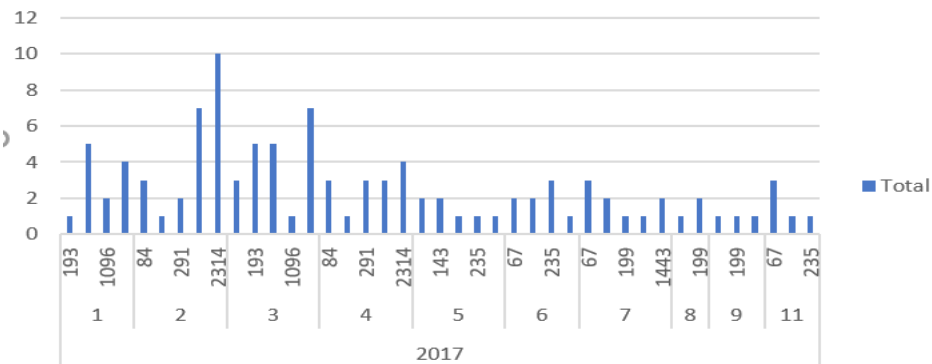
- Report 3

Total Compensation Per Month in 2017



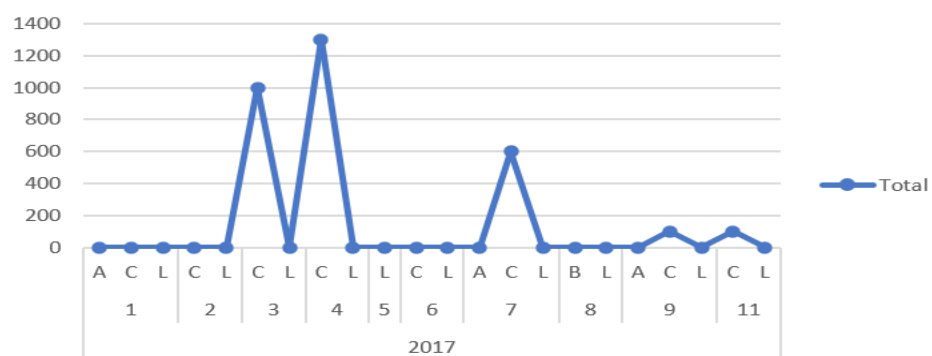
Report 4

Total Complaint Per Flight Per Month



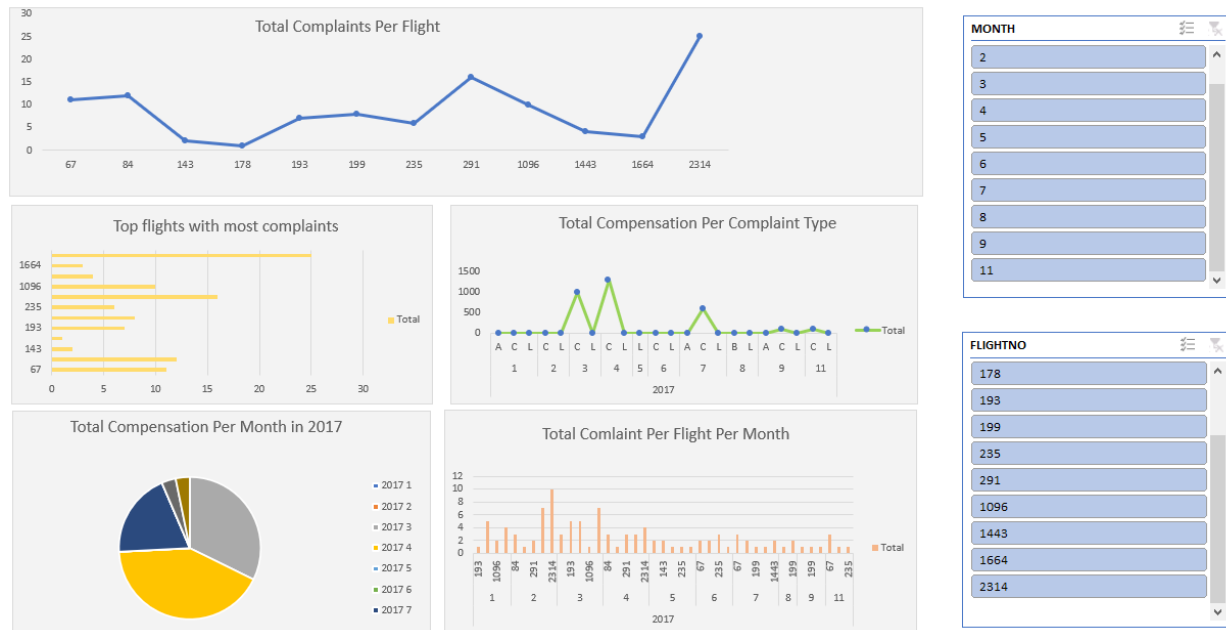
Report 5

Total Compensation Per Complaint Type



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 Inmon'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 Inmon'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 :

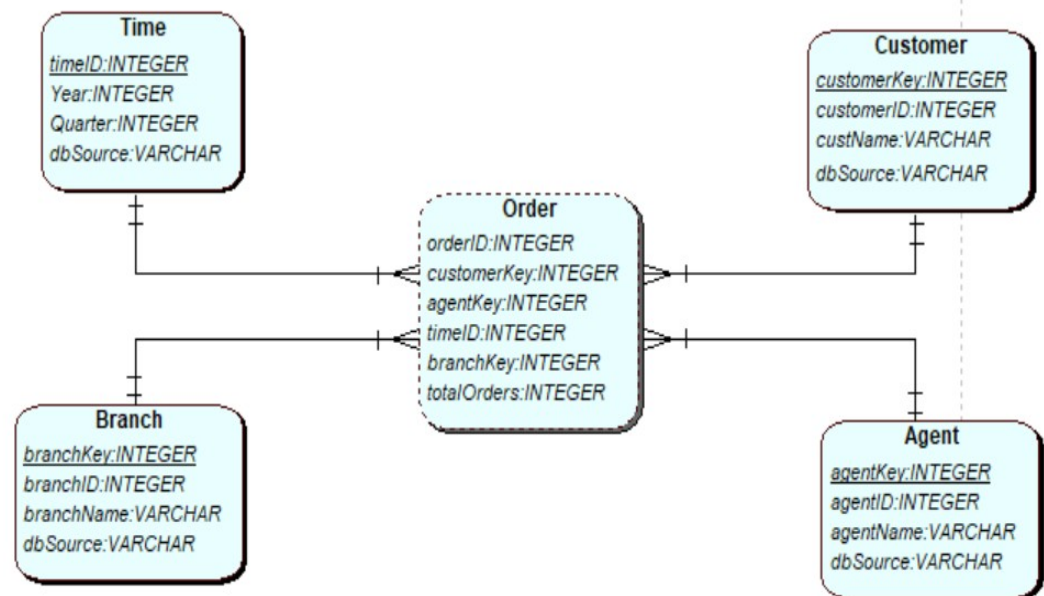
- In Kimball design, data isn't entirely integrated before reporting.
- As redundant data is added to database tables, data irregularities are most likely to occur.
- In the Kimball DW approach, the data warehouse model may be difficult to alter with any change in business needs.
- The model is business 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.

Entity Relationship Diagram



- 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)

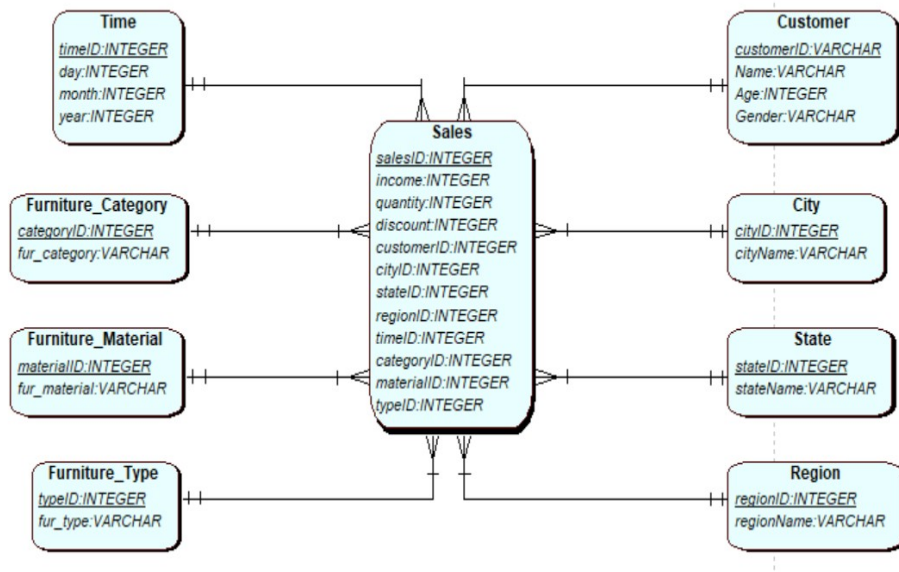
Customer (Age, Gender,

2. For each fact:

□ Produce the fact schema

STAR SCHEMA

Entity Relationship Diagram



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
```

