12/14/2019

|  |
| --- |
| Abhash| |



|  |  |
| --- | --- |
| 77190010 | advanced database |

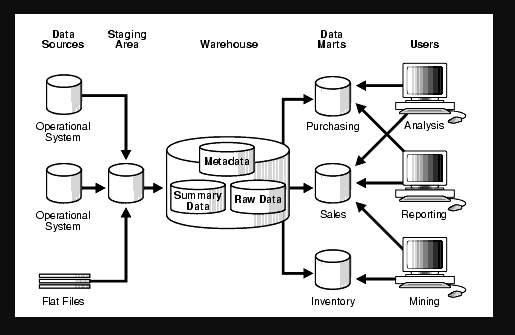
**Database Warehouse**

In simple terms data warehouse is a pool of data produced to support a decision making; it is also a repository of current and historical data of potential interest to man – agers throughout the organization. Data are usually structured to be available in a form ready for analytical processing activities like online analytical processing [OLAP], data mining, querying, reporting and other decision support applications. A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decision-making process.

**Database Warehouse Design**

**The Inmon Approach**

The Inmon approach to building a data warehouse begins with the corporate data model. This model identifies the key subject areas, and most importantly, the key entities the business operates with and cares about, like customer, product, vendor, etc. From this model, a detailed logical model is created for each major entity. For example, a logical model will be built for Customer with all the details related to that entity. There could be ten different entities under Customer. All the details including business keys, attributes, dependencies, participation, and relationships will be captured in the detailed logical model. The key point here is that the entity structure is built in normalized form. Data redundancy is avoided as much as possible. This leads to clear identification of business concepts and avoids data update anomalies. The next step is building the physical model. The physical implementation of the data warehouse is also normalized. This is what Inmon calls as a ‘data warehouse,’ and here is where the single version of truth for the enterprise is managed. This normalized model makes loading the data less complex, but using this structure for querying is hard as it involves many tables and joins. So, Inmon suggests building data marts specific for departments. The data marts will be designed specifically for Finance, Sales, etc., and the data marts can have de-normalized data to help with reporting. Any data that comes into the data warehouse is integrated, and the data warehouse is the only source of data for the different data marts. This ensures that the integrity and consistency of data is kept intact across the organization. Figure below shows the typical architecture of an Inmon data warehouse.



**Advantages of Inmon Approach**

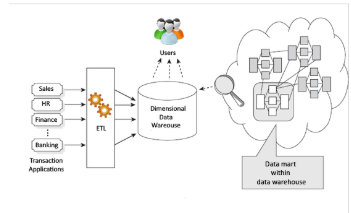
* The data warehouse truly serves as the single source of truth for the enterprise, as it is the only source for the data marts and all the data in the data warehouse is integrated.
* Data update anomalies are avoided because of very low redundancy. This makes ETL process easier and less prone to failure.
* The business processes can be understood easily, as the logical model represents the detailed business entities.
* Very flexible – As the business requirements change or source data changes, it is easy to update the data warehouse as one thing is in only one place.
* Can handle varied reporting needs across the enterprise.

**Disadvantages of Inmon Approach**

* The model and implementation can become complex over time as it involves more tables and joins.
* Need resources who are experts in data modeling and of the business itself. These type of resources can be hard to find and are often expensive.
* The initial set-up and delivery will take more time, and management needs to be aware of this.
* More ETL work is needed as the data marts are built from the data warehouse.
* A fairly large team of specialists need to be around to successfully manage the environment.

**The Kimball Approach**

The Kimball approach to building the data warehouse starts with identifying the key business processes and the key business questions that the data warehouse needs to answer. The key sources (operational systems) of data for the data warehouse are analyzed and documented. ETL software is used to bring data from all the different sources and load into a staging area. From here, data is loaded into a dimensional model. Here the comes the key difference: the model proposed by Kimball for data warehousing—the dimensional model—is not normalized. The fundamental concept of dimensional modeling is 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. Multiple star schemas will be built to satisfy different reporting requirements. So, how is integration achieved in the dimensional model? Here, Kimball proposes the concept of ‘conformed dimensions’. The key dimensions, like customer and product, that are shared across the different facts will be built once and be used by all the facts. This ensures that one thing or concept is used the same way across the facts. Another key artifact of the Kimball model is the ‘enterprise bus matrix’. This is the document where the different facts are listed vertically and the conformed dimensions are listed horizontally. Where ever the dimensions play a foreign key role in the fact, it is marked in the document. This serves as an anchoring document showing how the star schemas are built and what is left to build in the data warehouse. Figure below shows a typical Kimball data warehouse architecture.



**Advantages of Kimball Approach**

* Quick to set-up and build, and the first phase of the data warehousing project will be delivered quickly.
* The star schema can be easily understood by the business users and is easy to use for reporting. Most BI tools work well with star schema.
* The foot print of the data warehousing environment is small; it occupies less space in the database and it makes the management of the system fairly easier.
* The performance of the star schema model is very good. The database engine will perform a ‘star join’ where a Cartesian product will be created using all of the dimension values and the fact table will be queried finally for the selective rows. This is known to be a very effective database operation.
* A small team of developers and architects is enough to keep the data warehouse performing effectively.

**Disadvantages of Kimball Approach**

* The essence of the ‘one source of truth’ is lost, as data is not fully integrated before serving reporting needs.
* Redundant data can cause data update anomalies over time.
* Adding columns to the fact table can cause performance issues. This is because the fact tables are designed to be very deep. If new columns are to be added, the size of the fact table becomes much larger and will not perform well. This makes the dimensional model hard to change as the business requirements change.
* Cannot handle all the enterprise reporting needs because the model is oriented towards business processes rather than the enterprise as a whole.
* Integration of legacy data into the data warehouse can be a complex process.

**Difference between Inmon and Kimball Approach**

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Kimball** | **Inmon** |
| Business decision support requirements | Tactical | Strategic |
| Data integration requirements | Individual business requirements | Enterprise-wide integration |
| The structure of data | KPI, business performance measures, scorecards… | Data that meet multiple and varied information needs and non-metric data |
| Time constraint | Urgent needs for the first data warehouse | Longer time is allowed to meet business’ needs. |

**Star Schema**

Star schemas offer the simplest structure for organizing data into a data warehouse. The center of a star schema consists of one or multiple “fact tables” that index a series of “dimension tables.” To understand star schemas—and for that matter snowflake schemas—it's important to look at fact tables and dimension tables in depth.

The purpose of a star schema is to cull out numerical "fact" data relating to a business, and separate it from the descriptive, or “dimensional" data. Fact data will include information like price, weight, speed, and quantities—i.e., data in a numerical format. Dimensional data will include uncountable things like colors, model names, geographical locations, employee names, salesperson names, etc., along with the numerical information.

The fact data is organized into fact tables, and the dimensional data is organized into dimension tables. Fact tables are the points of integration at the center of the star schema in the data warehouse. They allow machine learning tools to analyze the data as a single unit, and they allow other business systems to access the data together. Dimension tables hold and manage the data—numerical and non numerical—which converges through fact tables that make up the data warehouse.

From a more technical perspective, fact tables keep track of numerical information related to different events. For example, they might include numeric values along with foreign keys that map to additional (descriptive and non numerical) information in the dimension tables. Getting even more technical, fact tables maintain a low level of granularity (or “detail”), which is to say, they record information at a more atomic level. This could lead to the buildup of many records within the fact table over time.

**Snowflake Schema**

The purpose of a snowflake schema is to normalize the de normalized data in a star schema. This solves the write command slow-downs and other problems typically associated with “star schemas.”

The snowflake schema is a “multi-dimensional” structure. At its core are fact tables that connect the information found in the dimension tables, which radiate outward like in the star schema. The difference is that the dimension tables in the snowflake schema divide themselves into more than one table. That creates the snowflake pattern.

Through this “snowflaking” method, the snowflake schema normalizes the dimension tables it connects with by (1) getting rid of “low cardinality” attributes (that appear multiple times in the parent table); and (2) turning the dimension tables into more than one table, until the dimension tables are completely normalized.

Like snowflake patterns in nature, the snowflake database becomes exceedingly complex. The schema can produce elaborate data relationships, where child tables have more than one parent table.

**OLTP VS OLAP**

OLTP (online transaction processing system) is a term used for transaction system, which is responsible for capturing and storing data related to day-to-day business functions such as ERP(Enterprise Resource Planning), CRM(Customer Relationship Management), SCM(Supply chain management), point of sale, and so forth. The OLTP system addresses a critical business need, automating daily business transactions and running real-time reports and routine analyses. But these systems are not designed for ad hoc analysis and complex queries that deal with a number of data items. OLAP, on the other hand, is designed to address this need by providing ad hoc analysis of organizational data much more effectively and efficiently. OLAP and OLTP rely heavily on each other: OLAP uses the data captures by OLTP, and OLTP automates the business processes that are managed by decisions supported by OLAP. Table below provides a multi-criteria comparison between OLTP and OLAP.

**A Comparison Between OLTP and OLAP**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **OLTP** | **OLAP** |
| Purpose  Data Source  Reporting  Resource Requirements  Execution Speed | To carryout day-to-day business functions.  Transaction Database (a normalized data repository primarily focused on efficiency and consistency).  Routine, periodic, narrow focused reports.  Ordinary relational database.  Fast (recording of business transactions and routine reports). | To support decision making and provide answers to business and management queries.  Data warehouse or data mart (a non normalized data repository primarily focused on completeness).  Ad hoc, multi-dimensional, broadly focused reports and queries.  Multiprocessor, large-capacity, specialized database.  Slow (resource intensive, complex, large-scale queries) |

**Task 1**

**Project Overview**

In simple terms data warehouse is a pool of data produced to support a decision making; it is also a repository of current and historical data of potential interest to man – agers throughout the organization. Data are usually structured to be available in a form ready for analytical processing activities like online analytical processing [OLAP], data mining, querying, reporting and other decision support applications. A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decision-making process.

The process of analyzing datasets to draw conclusion about the information they contain is known as data analytics, as the name suggests data warehouse is a repository of data from different data sources. The role of the database warehouse is it helps in better decision making, analyzing the large amount of historical data helps in better decision making. Data anomalies are the problem caused by poorly planned database since data warehouse provides data quality and data consistency which reduces the problem of data consistency.

Most organizations have a data warehouse and many have more than one. But the world has changed since the time those data warehouses were initially implemented. New data sources, new data types, new database technologies, new use cases, and new users of data combine to raise the question

Data warehouses are here for the long term. Much has been invested in building them and many people and business functions depend on them. But sustainability demands that we rethink the data warehouse. Data warehouse architecture can no longer stand alone. We must think purpose, placement, and positioning of the data warehouse in broader data management architecture.

Fly U is an airline company which keeps records of flights that leaves each airport, they record the planned departure and arrival of flights and the actual arrival and departure times of flights. They also keep information on which passengers are on each flight along with the lead and deputy pilot for the flight.

They keep records of complaints raised by the passengers against flights. The system provides information for the airline on flight delays which they use in their planning and customer complaints service. FlyU has asked us to take on a Data Mart project for their company to support the design, analysis and collection of information relating to their business. They wish to investigate three main key performance indicators:

i. Deliver a quality service.

ii. Ensure customer satisfaction

iii. Grow the company.

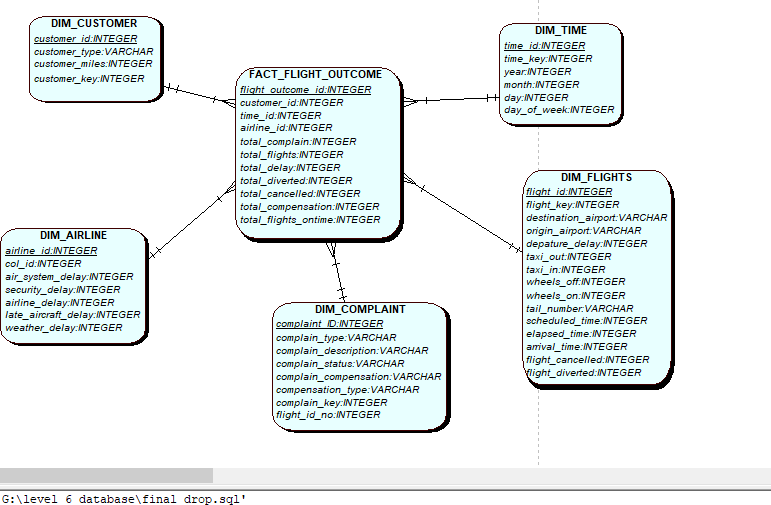
Among the above mentioned three key performance indicators, this Data Mart project focuses on building a data warehouse whose primary objective is to help FlyU to deliver a quality service. \

**Task 2**

*Identify 3-5 reports\* that your star schema will support*

* Number of flights which are delayed per year
* Number of cancelled flights per year
* Number of flights each year
* Number of minutes spend by airplane on air per year
* Number of flights delayed due to security reasons
* Number of flights arrived in time per year
* Number of Scheduled departure per year

*\*Document the star schema (SS) design model to support these reports – use QSEE*



**Star schema Design**

**Data dictionary 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Star schema table** | | | | **Attribute name** | | | | **Data Type** | | | | **Key** | | | | **DQ Source** | | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** |
| **DIM\_COMPLAINT** | | | | complaint\_id | | | | INTEGER | | | | Yes | | | | Automatically generated as primary key | | | | n/a | | | | n/a | | | | Create a sequence complaint\_seq to support the generation of primary keys. |
| complain\_type | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.type | | | | Some complaint types are missing. | | | | n/a |
| complain\_description | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.description | | | | Some description is missing. | | | | n/a |
| complain\_status | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.complaint\_status | | | | No Data quality issues | | | | n/a |
| complain\_compensation | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.compensation\_amnt | | | | No data quality issues. | | | | n/a |
| compensation\_type | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.compensation\_type | | | | Some compensation type data is missing. | | | | n/a |
| complain\_key | | | | INTEGER | | | | No | | | | Automatically generated as a surrogate key | | | | n/a | | | | n/a | | | | Create a sequence complaint\_id\_seq to support the generation of primary keys. |
| flight\_id\_no | | | | INTEGER | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.flight\_id\_no | | | | No data quality issues. | | | | n/a |
| Definition: | | | | The complaint dimension table holds data from the complaint table as given in the FlyU\_flights\_5\_sept.sql script. The table includes data for flight\_id\_no, complaint\_description, complaint\_status etc. | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: | | | |  | | | | | | | | | | | | | | | | | | | | | | | | |
| **Star schema table** | | | | **Attribute name** | | | | **Data Type** | | | | **Key** | | | | **DQ Source** | | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** |
| **DIM\_TIME** | | | | time\_id | | | | INTEGER | | | | Yes | | | |  | | | | n/a | | | | n/a | | | | Create a sequence time\_seq to support the generation of primary key. |
| time\_key | | | | INTEGER | | | | No | | | | Automatically generated as a surrogate key | | | | n/a | | | | n/a | | | | Create a sequence time\_id\_seq to support the generation of surrogate keys. |
| year | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.year and Flight\_2016\_2000\_rows.year | | | | No data quality issues. | | | | n/a |
| month | | | | INTEGER | | | | No | | | | Flight\_2016\_2000\_rows.xlsx | | | | Flight\_2016\_2000\_rows.month | | | | No data quality issues. | | | | n/a |
| day | | | | INTEGER | | | | No | | | | Flight\_2016\_2000\_rows.xlsx | | | | Flight\_2016\_2000\_rows.day | | | | No data quality issues. | | | | n/a |
| day\_of\_week | | | | INTEGER | | | | No | | | | Flight\_2016\_2000\_rows.xlsx | | | | Flight\_2016\_2000\_rows.day\_of\_week | | | | No data quality issues. | | | | n/a |
| Definition: | | | | The time dimension table holds the intervals of time for which the data will be held. It is held at year level meaning now reports are available at a lower granularity. | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: | | | |  | | | | | | | | | | | | | | | | | | | | | | | | |
| **Star schema table** | | | | **Attribute name** | | | | **Data Type** | | | | **Key** | | | | **DQ Source** | | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** |
| **DIM\_CUSTOMER** | | | | customer\_id | | | | INTEGER | | | | Yes | | | | Automatically generated as primary key | | | | n/a | | | | n/a | | | | Create a sequence customer\_seq to support the generation of primary key. |
| customer\_type | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.customer\_type | | | | No data quality issues. | | | | n/a |
| customer\_miles | | | | VARCHAR | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | FlyU\_flights\_5\_sept.customer\_miles | | | | No data quality issues | | | | n/a |
| customer\_key | | | | INTEGER | | | | No | | | | Automatically generated as a surrogate key | | | | n/a | | | | n/a | | | | Create a sequence customer\_id\_seq to support the generation of surrogate key. |
| Definition: | | | | The customer dimension table holds the data from FlyU\_flights\_5\_sept.sql script which contains data for customer\_type and customer\_miles. | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: | | | |  | | | | | | | | | | | | | | | | | | | | | | | | |
| **Star schema table** | | | **Attribute name** | | **Data Type** | | | | **Key** | | | | | **DQ Source** | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** | | | |
| **DIM\_AIRLINE** | | | airline\_id | | INTEGER | | | | Yes | | | | | Automatically generated as a primary key | | | n/a | | | | n/a | | | | Create a sequence airline\_seq to support the generation of primary keys | | | |
| col\_id | | INTEGER | | | | No | | | | | Automatically generated as surrogate key | | | n/a | | | | n/a | | | | Create a sequence col\_id\_seq to support the generation of primary keys | | | |
| air\_system\_delay | | INTEGER | | | | No | | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | 2015\_3000.AIR\_SYSTEM\_DELAY and Flight\_2016\_2000\_rows. AIR\_SYSTEM\_DELAY | | | | Some data is missing from the column. | | | | n/a | | | |
| security\_delay | | INTEGER | | | | No | | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | 2015\_3000.AIR\_SECURITY\_DELAY and Flight\_2016\_2000\_rows. SECURITY\_DELAY | | | | Some data is missing from the column. | | | | n/a | | | |
| airline\_delay | | INTEGER | | | | No | | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | 2015\_3000.AIR\_SECURITY\_DELAY and Flight\_2016\_2000\_rows. SECURITY\_DELAY | | | | Some data is missing from the column. | | | | n/a | | | |
| late\_aircraft\_delay | | INTEGER | | | | No | | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | 2015\_3000.LATE\_AIRCRAFT\_DELAY and Flight\_2016\_2000\_rows. LATE\_AIRCRAFT\_DELAY | | | | Some data is missing from the column. | | | | n/a | | | |
| weather\_delay | | INTEGER | | | | No | | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | 2015\_3000.WEATHER\_DELAY and Flight\_2016\_2000\_rows.WEATHER\_DELAY | | | | Some data is missing from the column. | | | | n/a | | | |
| Definition: | | | The airline dimension table holds data from 2015\_3000.xlsx and Flight\_2016\_2000\_rows.xlsx data sources which contains data for weather\_delay, late\_aircraft\_delay, airline\_delay, security\_delay etc. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: | | |  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Star schema table** | | **Attribute name** | | | | | **Data Type** | | | | **Key** | | | | **DQ Source** | | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** | |
|  | | flight\_id | | | | | INTEGER | | | | Yes | | | | Automatically generated as a primary key | | | | n/a | | | | n/a | | | | Create a sequence flights\_seq to support the generation of primary keys. | |
| flight\_key | | | | | INTEGER | | | | No | | | | Automatically generated as a surrogate key | | | | n/a | | | | n/a | | | | Create a sequence flights\_seq to support the generation of primary keys. | |
| destination\_airport | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.DESTINATION\_AIRPORT and Flight\_2016\_2000\_rows.DESTINATION\_AIRPORT | | | | No data quality issues. | | | | n/a | |
| origin\_airport | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.ORIGIN\_AIRPORT and Flight\_2016\_2000\_rows.ORIGIN\_AIRPORT | | | | No data quality issues. | | | | n/a | |
| departure\_delay | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.DEPARTURE\_DELAY and Flight\_2016\_2000\_rows.DEPARTURE\_DELAY | | | | Some data is missing. | | | | n/a | |
| taxi\_out | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.TAXI\_OUT and Flight\_2016\_2000\_rows.TAXI\_OUT | | | | Some data is missing. | | | | n/a | |
| taxi\_in | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.TAXI\_IN and Flight\_2016\_2000\_rows.TAXI\_IN | | | | Some data is missing. | | | | n/a | |
| wheels\_off | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.WHEELS\_OFF and Flight\_2016\_2000\_rows.WHEELS\_OFF | | | | Some data is missing. | | | | n/a | |
| wheels\_on | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.WHEELS\_ON and Flight\_2016\_2000\_rows.WHEELS\_ON | | | | Some data is missing. | | | | n/a | |
| tail\_number | | | | | VARCHAR | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.TAIL\_NUMER and Flight\_2016\_2000\_rows.TAIL\_NUMBER | | | | Some data is missing | | | | n/a | |
| scheduled\_time | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.SCHEDULED\_TIME and Flight\_2016\_2000\_rows.SCHEDULED\_TIME | | | | No data is missing | | | | n/a | |
| elapsed\_time | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.ELAPSED\_TIME and Flight\_2016\_2000\_rows.ELAPSED\_TIME | | | | Some data is missing | | | | n/a | |
| arrival\_delay | | | | | INTEGER | | | | No | | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | 2015\_3000.ARRIVAL\_DELAY and Flight\_2016\_2000\_rows.ARRIVAL\_DELAY | | | | Some data is missing | | | | n/a | |
| flight\_cancelled | | | | | INTEGER | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | 2015\_3000.CANCELLED and Flight\_2016\_2000\_rows.CANCELLED | | | | All data are null. | | | | n/a | |
| flight\_diverted | | | | | INTEGER | | | | No | | | | FlyU\_flights\_5\_sept.sql | | | | 2015\_3000.DIVERTED and Flight\_2016\_2000\_rows.DIVERTED | | | | All data are null. | | | | n/a | |
| Definition: | | The flights dimension table holds data from 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx and FlyU\_flights\_5\_sept.sql data sources which contains data for departure\_delay, wheels\_off, wheels\_on,tail\_number, elapsed\_time etc. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: | |  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Star schema table** | **Attribute name** | | | | | **Data Type** | | | | **Key** | | | **DQ Source** | | | | | **Data source field (mapping)** | | | | **Data quality Issues** | | | | **Transformation** | | |
| **FACT\_FLIGHT\_OUTCOME** | flight\_outcome\_id | | | | | INTEGER | | | | Yes | | | Automatically generated as a primary key | | | | | n/a | | | | n/a | | | | Create a sequence outcome\_seq to support the generation of primary key. | | |
| customer\_id | | | | | INTEGER | | | | No | | | Automatically generated as a foreign key | | | | | n/a | | | | n/a | | | | Create a sequence customer\_seq to support the generation of primary key. | | |
| complaint\_id | | | | | INTEGER | | | | No | | | Automatically generated as a primary key | | | | | n/a | | | | n/a | | | | Create a sequence complaint\_seq to support the generation of primary key. | | |
| time\_id | | | | | INTEGER | | | | No | | | Automatically generated as a primary key | | | | | n/a | | | | n/a | | | | Create a sequence time\_seq to support the generation of primary key. | | |
| flight\_id | | | | | INTEGER | | | | No | | | Automatically generated as a primary key | | | | | n/a | | | | n/a | | | | Create a sequence flights\_seq to support the generation of primary key. | | |
| airline\_id | | | | | INTEGER | | | | No | | | Automatically generated as a primary key | | | | | n/a | | | | n/a | | | | Create a sequence airline\_seq to support the generation of primary key. | | |
|  | total\_complaint | | | | | INTEGER | | | | No | | | FlyU\_flights\_5\_sept.sql | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_flights | | | | | INTEGER | | | | No | | | FlyU\_flights\_5\_sept.sql | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_delay | | | | | INTEGER | | | | No | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_diverted | | | | | INTEGER | | | | No | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_cancelled | | | | | INTEGER | | | | No | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_compensation | | | | | INTEGER | | | | No | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | |  | | | | Some data missing. | | | | n/a | | |
|  | total\_flights\_ontime | | | | | INTEGER | | | | No | | | 2015\_3000.xlsx, Flight\_2016\_2000\_rows.xlsx | | | | |  | | | | Some data missing. | | | | n/a | | |
| Definition: | The fact\_flight\_outcome fact table contains data from all the given sources. It contains the foreign key of all the dimension table in the star schema and also contains data like total\_flights\_ontime, total\_cancelled, total compensation etc. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Notes: |  | | | | | | | | | | | | | | | | | | | | | | | | | | | |

:

**Data dictionary 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source (s)** | **Field Name** | | **Data Type** | **Key** | | **DQ Source** | **Data Quality Check** | | **Data quality Issues** | | | **Action Note** | |
| 2015\_3000 | **Flight\_id** | | Integer | No | | 2015\_3000 | Consistent | | No inconsistency found | | |  | |
| Flight\_2016 | Integer | No | | Flight\_2016 | Consistent | | No inconsistency found | | |  | |
| Definition: | Flight \_id holds the value of flight number | | | | | | | | | | | | |
| Notes: | Flight \_id column exist in both of our data sets. | | | | | | | | | | | | |
| 2015\_3000 | **Destination\_airport** | | Varchar2 | No | | 2015\_3000 | Consistent | | No inconsistency found | | |  | |
| Flight\_2016 | Varchar2 | No | | Flight\_2016 | Consistent | | No inconsistency found | | |  | |
| Definition: | Column that holds vales of destination airport | | | | | | | | | | | | |
| Notes: | Destination\_airportColumn exist in both data sets | | | | | | | | | | | | |
| 2015\_3000 | **origin\_airport** | Varchar2 | | | No | | | 2015\_3000 | | Consistent | No inconsistency found | |  |
| Flight\_2016 | Varchar2 | | | No | | | Flight\_2016 | | Consistent | No inconsistency found | |  |
| Definition | Column holds the value of the origin of the flight | | | | | | | | | | | | |
| Notes | Column exists in both sets | | | | | | | | | | | | |
| 2015\_3000 | departure\_delay | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric value | | Add default value |
| Definition | Column holds the value of delay before the departure | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | taxi\_out | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Definition | Column holds the value of taxi time before wheel off | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | Taxi\_in | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Definition | Column holds the value of time of taxi in | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | wheels\_off | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has nonnumeric or empty values | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has nonnumeric or empty values | | Add default value |
| Definition | Column holds the value of time of wheel off | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | Wheels\_on | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Definition | Column holds the value of time of wheel on | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | tail\_number | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has Empty string | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has empty String | | Add default value |
| Definition | Column holds the value of the plane tail number | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | scheduled\_time | Integer | | | No | | | 2015\_3000 | | Consistent | No inconsistency found | |  |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Consistent | No inconsistency found | |  |
| Definition | Column holds the value of time of Scheduled Time | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | elapsed\_time | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Definition | Column holds the value of time of Elasped Time | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | arrival\_delay | Integer | | | No | | | 2015\_3000 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Inconsistent | Has non numeric or empty value | | Add default value |
| Definition | Column holds the value of time of Elasped Time | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | flight\_cancelled | Integer | | | No | | | 2015\_3000 | | Consistent | No inconsistency found | |  |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Consistent | No inconsistency found | |  |
| Definition | Column holds the value of Flight Cancalled | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |
| 2015\_3000 | flight\_diverted | Integer | | | No | | | 2015\_3000 | | Consistent | No inconsistency found | |  |
| Flight\_2016 | Integer | | | No | | | Flight\_2016 | | Consistent | No inconsistency found | |  |
| Definition | Column holds the value of Flight diverted | | | | | | | | | | | | |
| Note | Column exists in both Column | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DIM\_CUSTOMER | | | | | | | |
| Data Source (s) | Field Name | Data Type | Key | DQ Source | Data Quality Check | Data quality Issues | Action Note |
| FlyU\_flights\_5\_sept | customer\_id | Interger | yes | Customer | Consistent | --- | --- |
| customer\_type | varchar | no | Customer | consistent | --- | --- |
| Customer\_miles | integer |  |  | consistent | --- | --- |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **DIM\_TIME** | | | | | | | |
| **Data Source (s)** | **Field Name** | **Data Type** | **Key** | **DQ Source** | **Data Quality Check** | **Data quality Issues** | **Action Note** |
| FlyU\_flights\_5\_sept  2015\_3000.XLXS  Flight\_2016\_2000.XLXS | **time\_id** | Integer | yes | Customer | Consistent | --- | --- |
| **Day\_of \_week** | integer | no | FLYU\_flights  2015\_3000.XLXS  Flight\_2016\_2000.XLXS | inconsistent | Day of week not available in 2015\_3000.XLSX | 2015\_3000.XLSX should have Day\_of week as a column |
| **Day** | integer |  | FLYU\_flights  2015\_3000.XLXS  Flight\_2016\_2000.XLXS | inconsistent | Day not available in 2015\_3000.XLSX | 2015\_3000.XLSX should have Day as a column |
| **month** | integer |  | FLYU\_flights  2015\_3000.XLXS  Flight\_2016\_2000.XLXS | inconsistent | Month not available in 2015\_3000.XLSX | 2015\_3000.XLSX should have month week as a column |
| **year** | integer |  | FLYU\_flights  2015\_3000.XLXS  Flight\_2016\_2000.XLXS | consistent | --- | --- |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source (s)** | **Field Name** | **Data Type** | **Key** | **DQ Source** | **Data Quality Check** | **Data quality Issues** | **Action Note** |
| FLIGHT\_2015 | **col\_id** | Integer | No | FLIGHT\_2015 | consistency | No consistency found |  |
| Definition | Col\_id holds the value | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015 | **air\_system\_delay** | Integer  Integer | No  No | Flight\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Flight\_2016 |
| Definition | Column hold the value of time of air system delay | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **security\_delay** | Integer  integer | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of security delay | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **airline\_delay** | Integer  Integer | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of arline delay | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **late\_aircraft\_delay** | Integer  Integer | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of late aircraft delay | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **weather\_delay** | Integer  Integer | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of weather delay | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Source (s)** | **Field Name** | **Data Type** | **Key** | **DQ Source** | **Data Quality Check** | **Data quality Issues** | **Action Note** |
| FLIGHT\_2015 | **Complain\_type** | Integer | No | FLIGHT\_2015  FLIGHT\_2016 | consistency | No consistency found |  |
| FLIGHT\_2016 |
| Definition | Column holds the value of complain type | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015 | **Complain\_description** | varchar | No  No | Flight\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Flight\_2016 |
| Definition | Column hold the value of time of complain description | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **Complain\_status** | varchar  varchar | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of complain status | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **Complain\_compensation** | varchar  varchar | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of complain compensation | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |
| FLIGHT\_2015  Flight\_2016 | **Flight\_id\_no** | Integer  Integer | No  No | FLIGHT\_2015  Flight\_2016 | Inconsistency  inconsistency | Has empty value | Add default value |
| Definition | Column hold the value of time of flight Id no | | | | | | |
| Notes | Column exists in both of our data sets | | | | | | |

\*Discuss for each table any design decisions for example: your decisions on granularity of data

**Granularity of Data**

Granularity can be defined as the lowest level of information or the level of detail considered during the storage of data. For example, the performance of an employee is considered as high level of granularity where as daily performance of employee is considered to be low level of granularity.

**Time Dimension Table**

|  |  |  |
| --- | --- | --- |
| **DIM\_TIME** | | |
| Attribute Name | Data Type | Key |
| time\_id | INTEGER | PRIMARY |
| time\_key | INTEGER |  |
| year | INTEGER |  |
| month | INTEGER |  |
| day | INTEGER |  |
| day\_of\_week | INTEGER |  |

**Table 1: Time Dimension Table**

The time dimension table is granulated to store information on a monthly basis. For an airline company like FlyU keeping track of flights on a monthly basis is more sensible. In order to analyze incoming flights on a monthly basis and to support reports such as the number of delayed flights per month or number of cancelled flights per month, the time dimension table is granulated to year, month, day and day of the week.

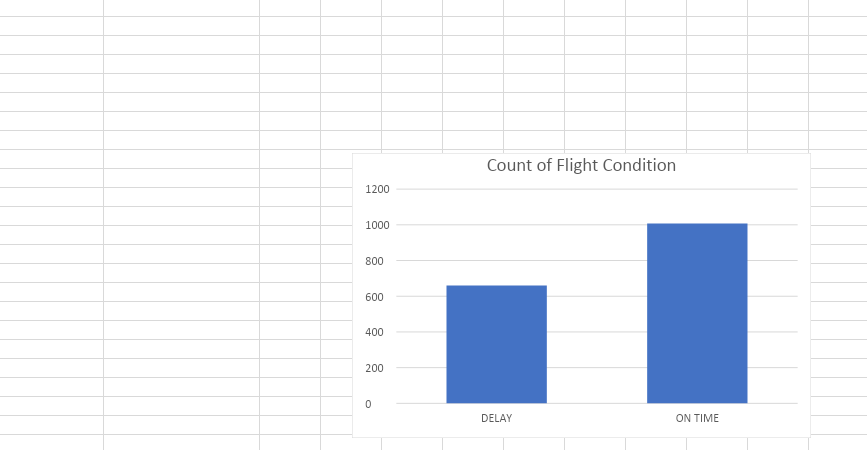
**Flight Outcome Fact Table**

|  |  |  |
| --- | --- | --- |
| **FACT\_FLIGHT\_OUTCOME** | | |
| Attribute Name | Data Type | Key |
| flight\_outcome\_id | INTEGER | PRIMARY |
| customer\_id | INTEGER | FOREIGN KEY |
| complaint\_id | INTEGER | FOREIGN KEY |
| time\_id | INTEGER | FOREIGN KEY |
| flight\_id | INTEGER | FOREIGN KEY |
| airline\_id | INTEGER | FOREIGN KEY |
| total\_complaint | INTEGER |  |
| total\_flights | INTEGER |  |
| total\_delay | INTEGER |  |
| total\_diverted | INTEGER |  |
| total\_cancelled | INTEGER |  |
| total\_compensation | INTEGER |  |
| total\_flights\_ontime | INTEGER |  |

**Table 2: Flight Outcome Fact Table**

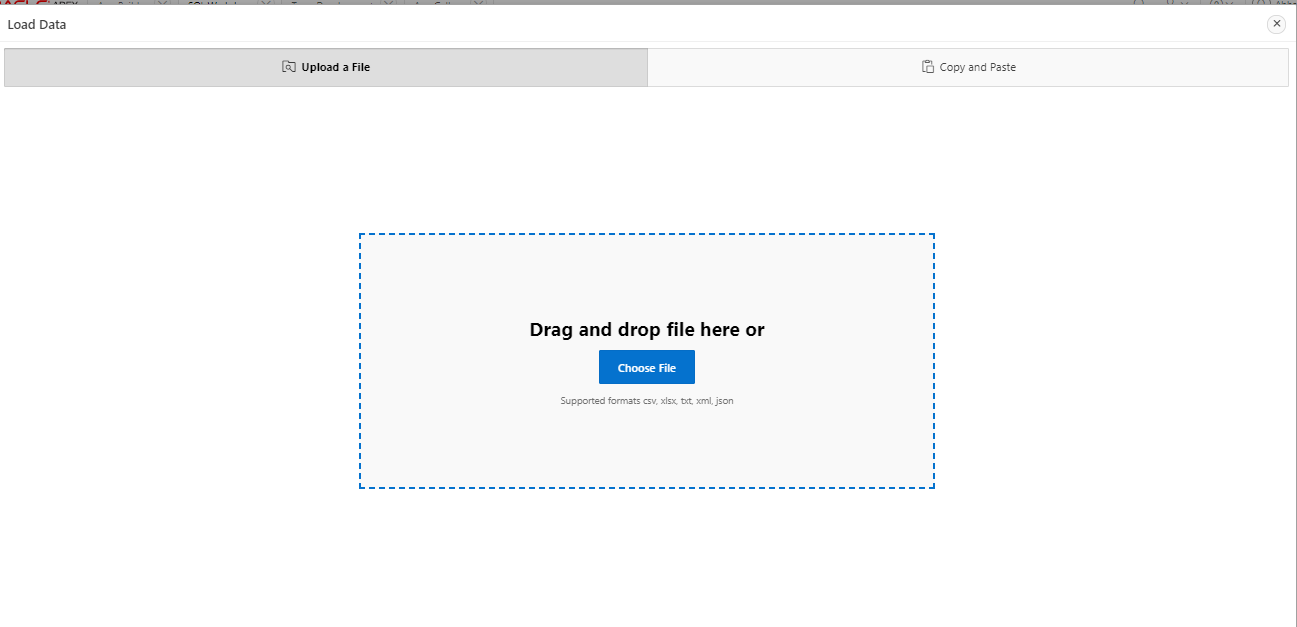
The fact table stores the records for each flight with the total flights, total flights delayed, total flights cancelled, total flights diverted, total compensation paid to customers and the total number of flights that arrived on schedule.

\*Select one of the reports\* you have suggested. Illustrate the expected data in the star schema to support the report - use Excel (or oracle or similar) to do this and add a few rows.

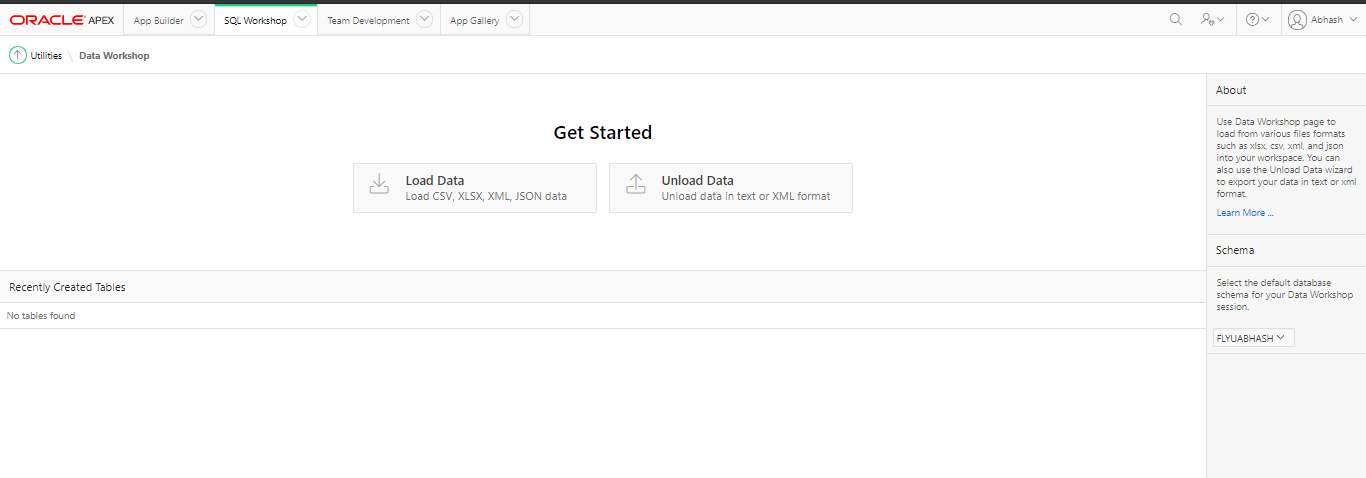
****

**Task 3**

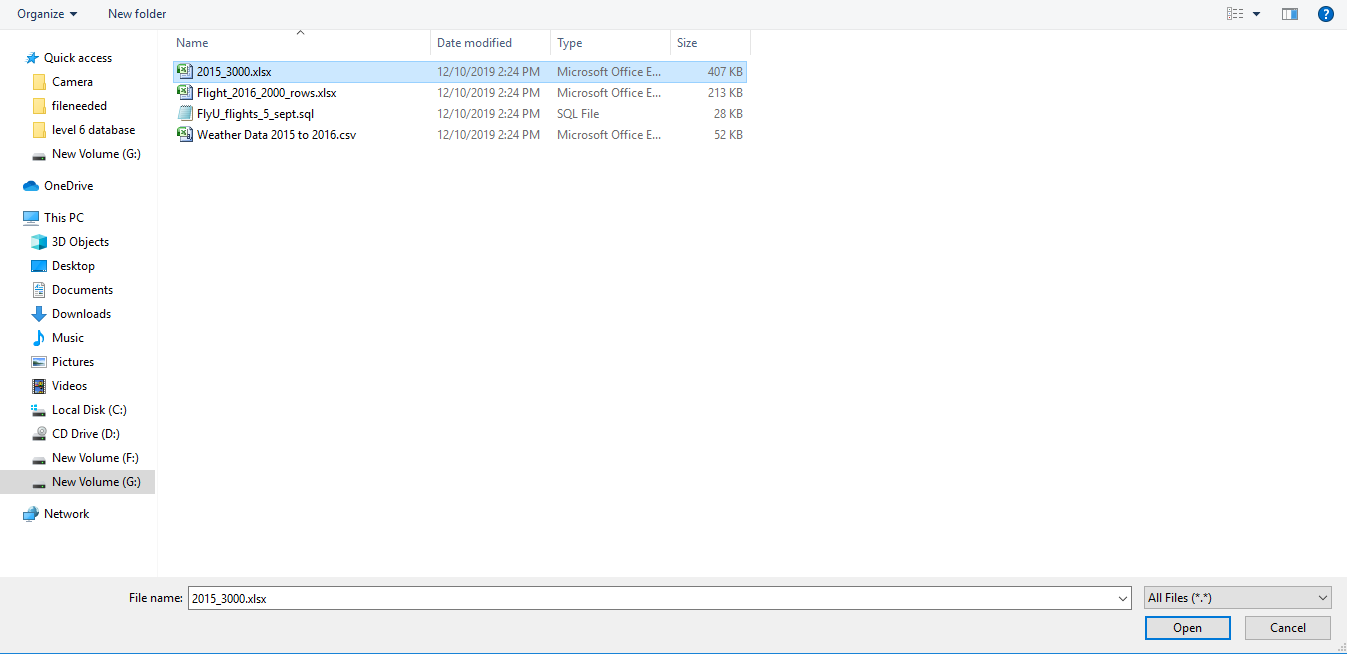
**Collect the data you require to populate the star schema (source data for DM) into the data staging area.**

****

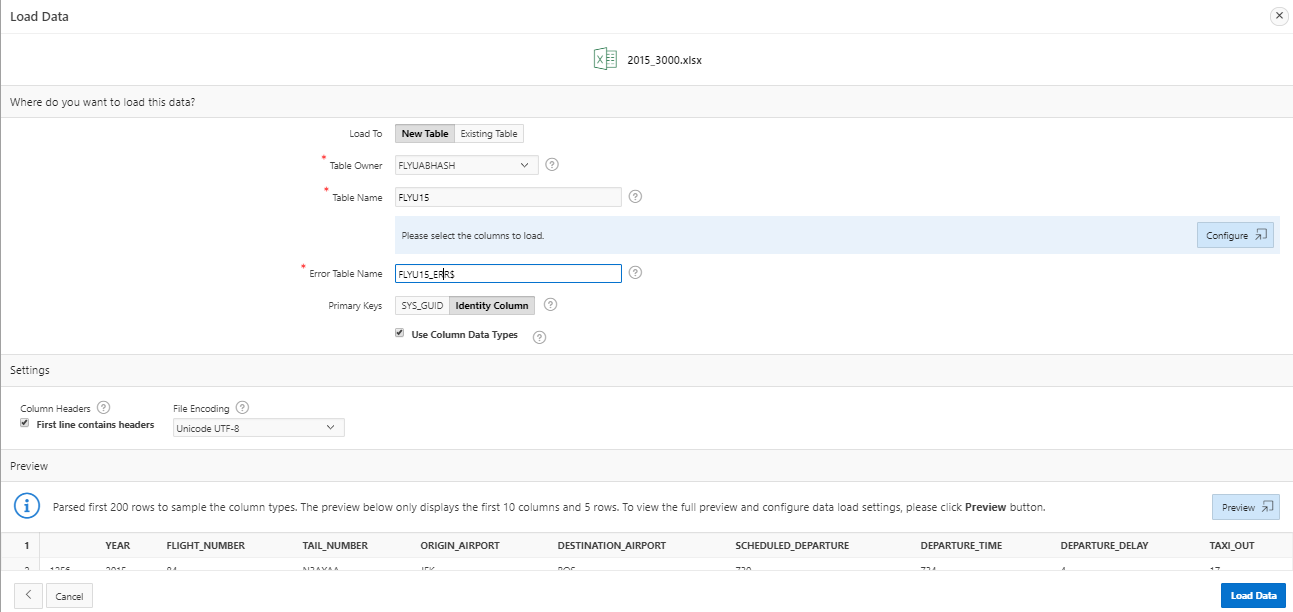
**Uploading the file in apex oracle**

****

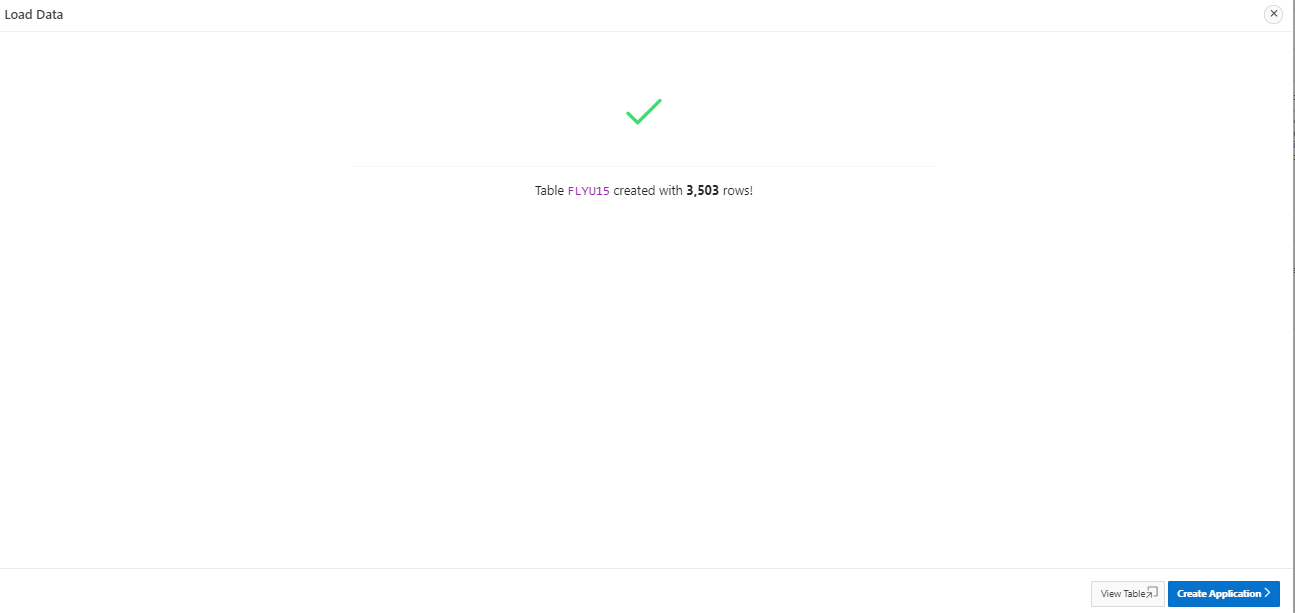
**Loading file in apex oracle**

****

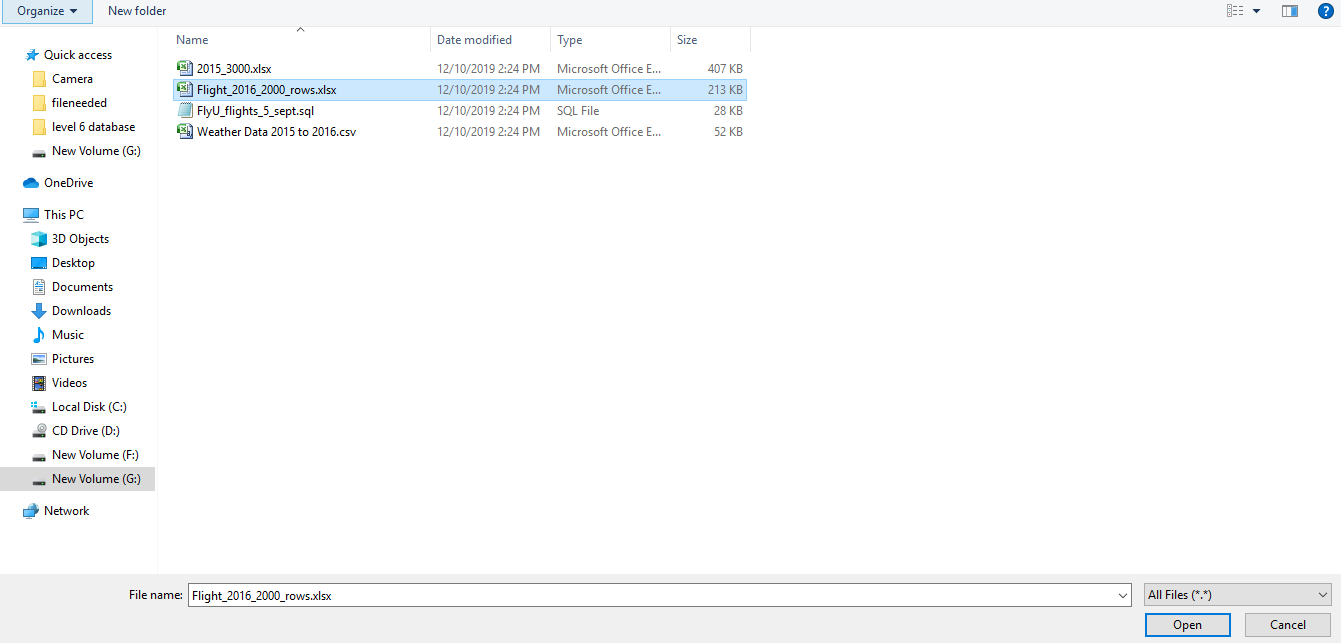
**Uploading 2015\_3000.xlxs file in apex oracle**

****

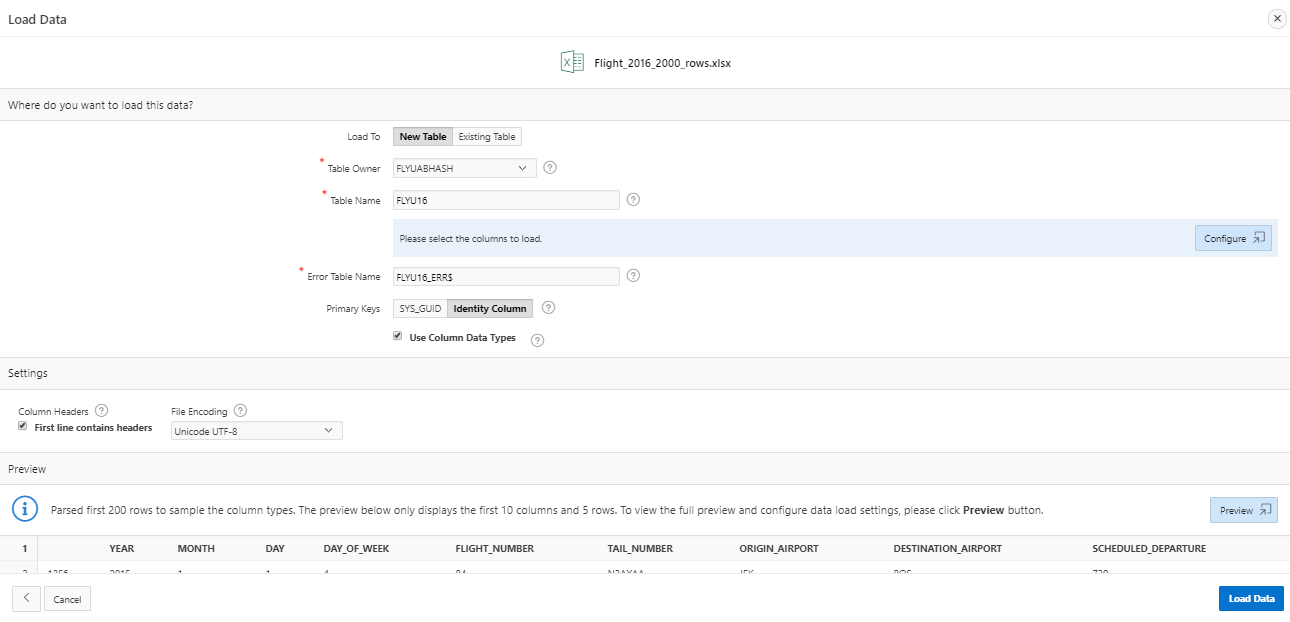
**Giving table name**

****

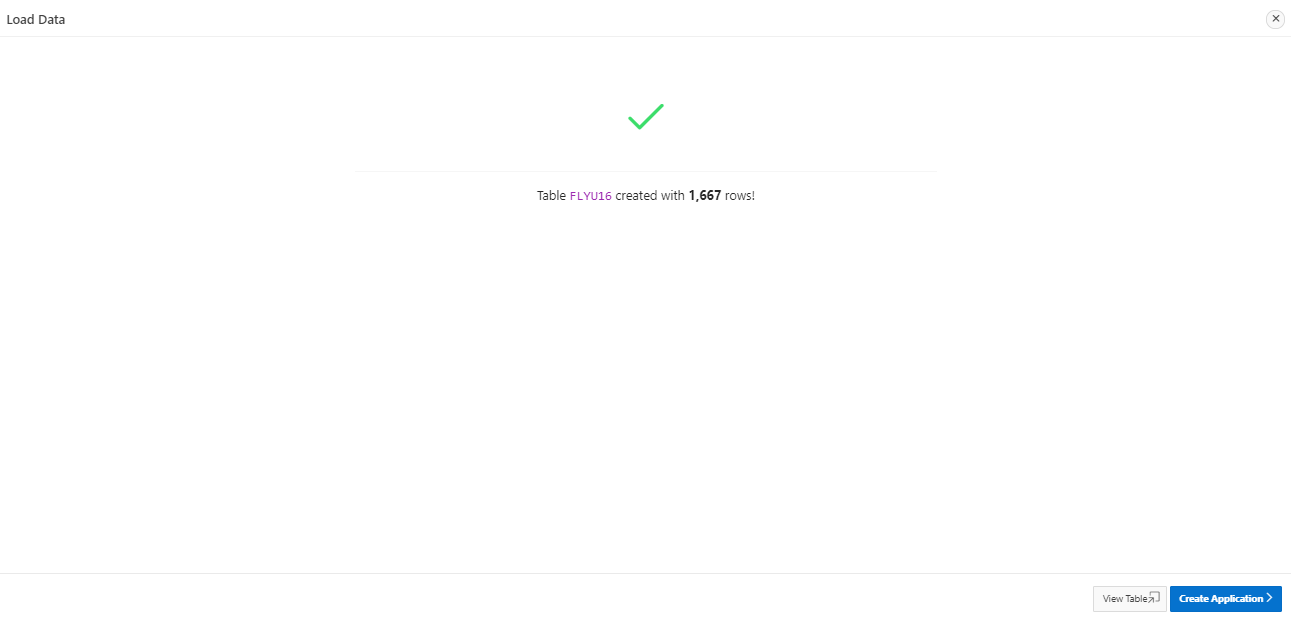
**The above screenshot shows that the table has been created**

****

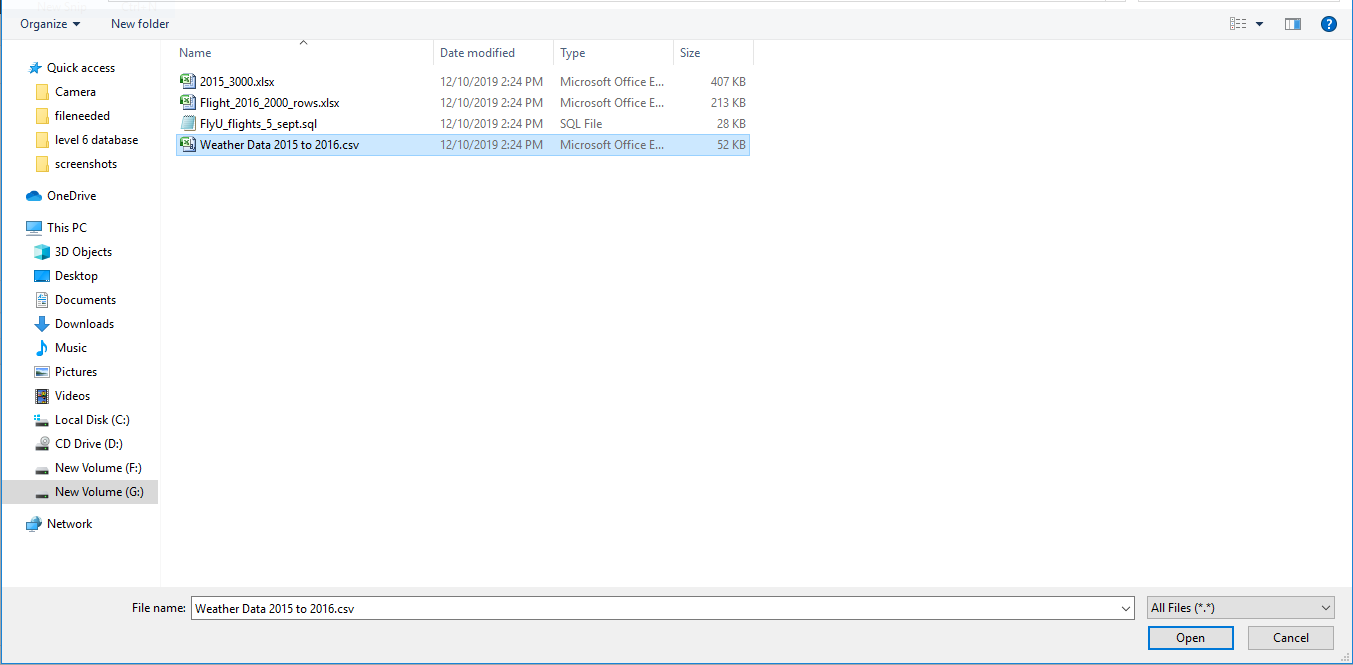
**Choosing Flight\_2016\_2000\_rows.xlsx file to load**

****

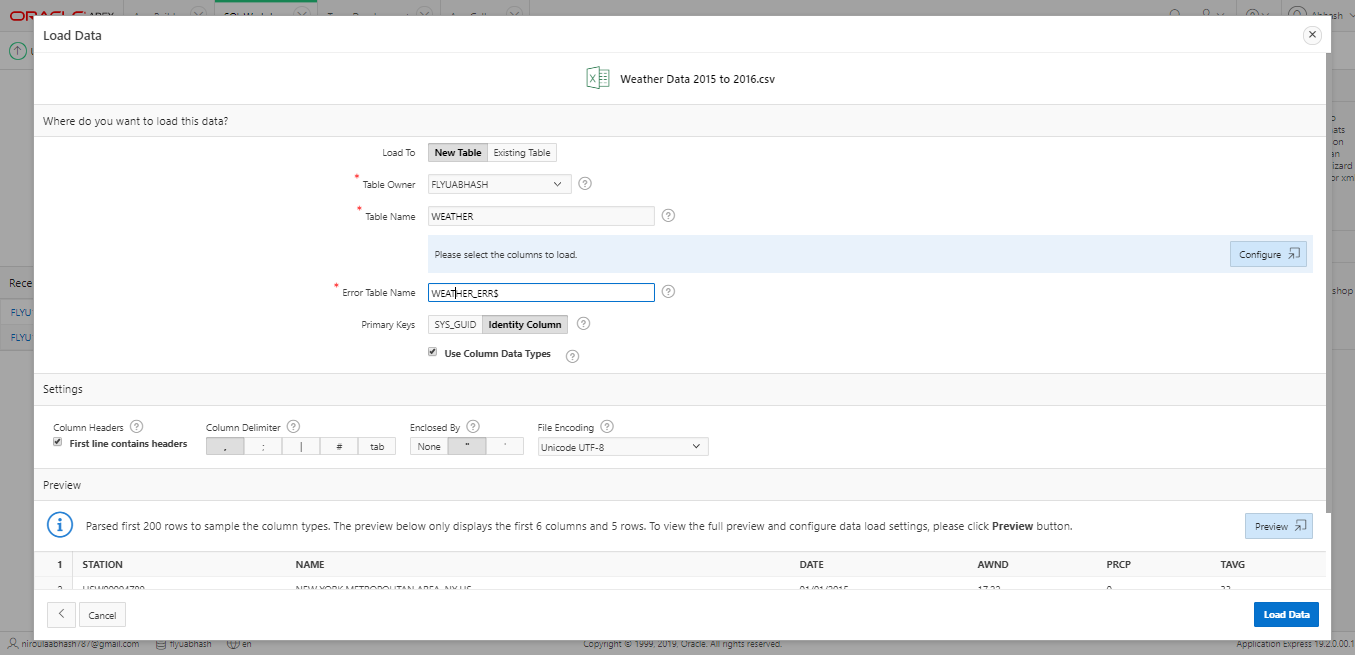
**Giving table name**

****

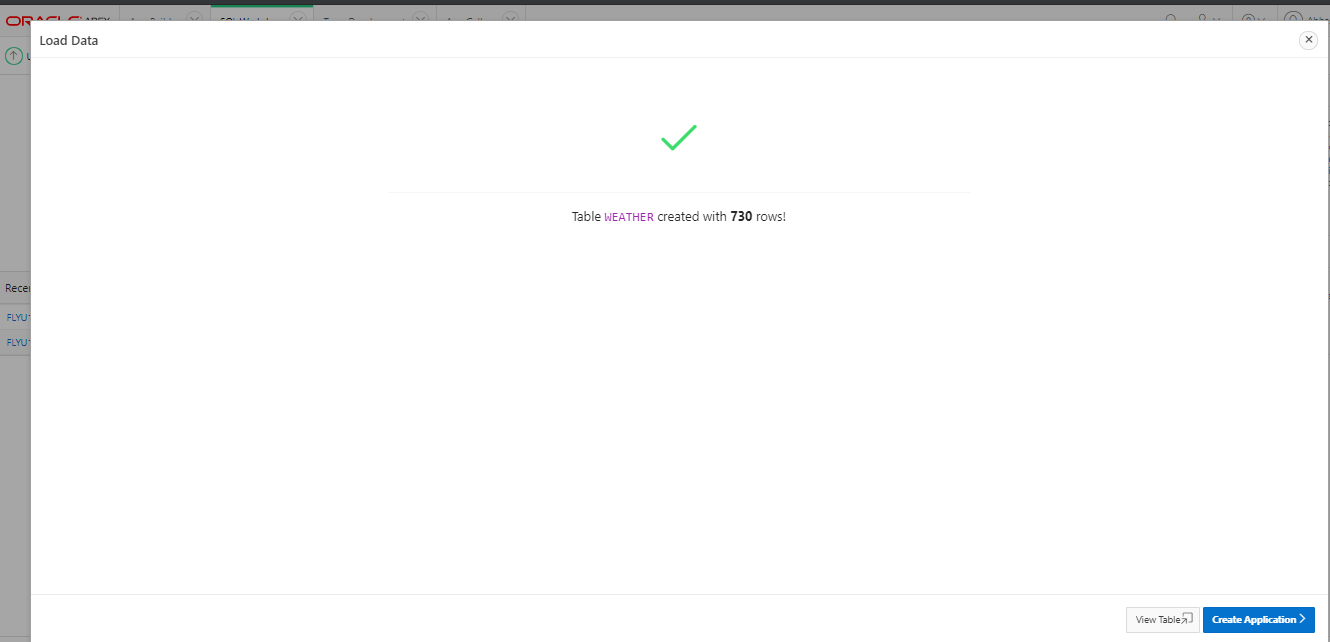
**The above screenshot shows that table has been created**

****

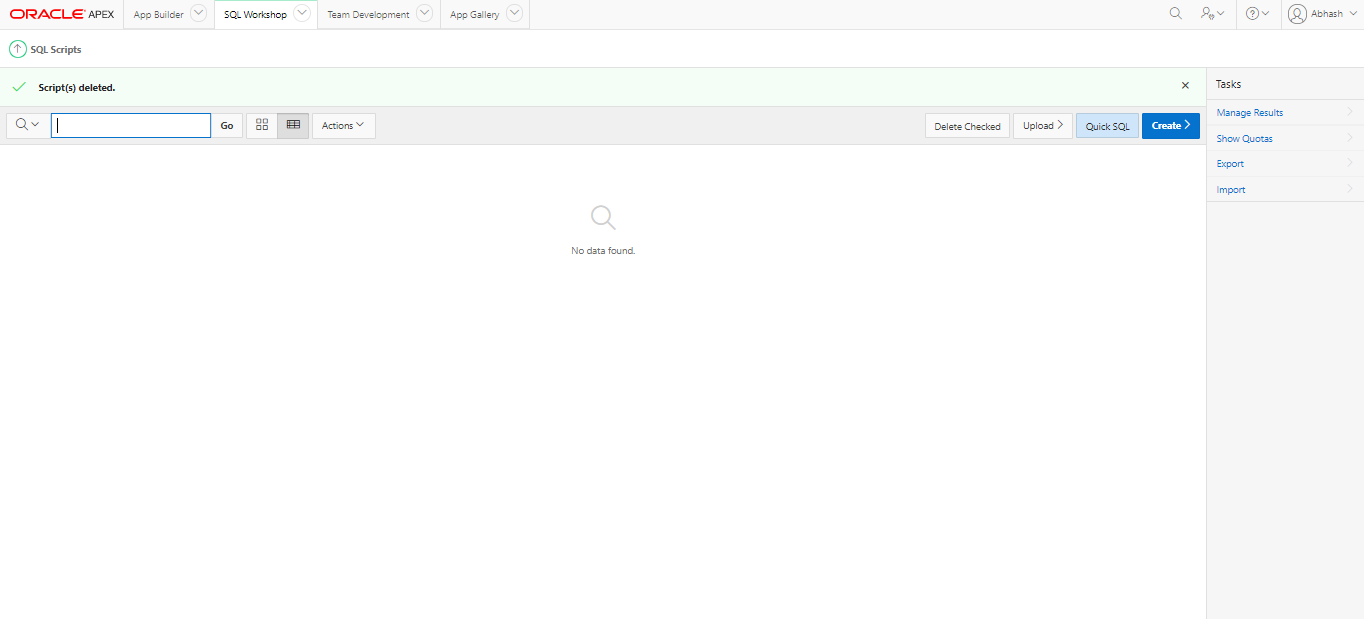
**Loading Weather Data 2015 to 2016.csv file**

****

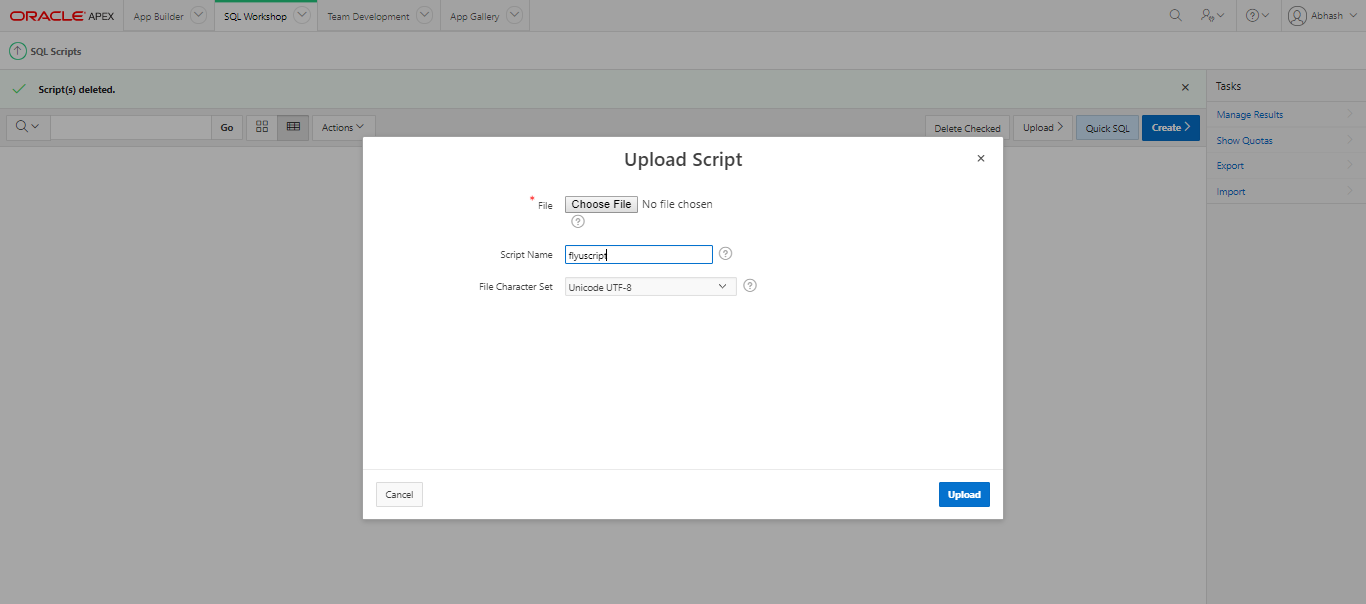
**Giving table name**

****

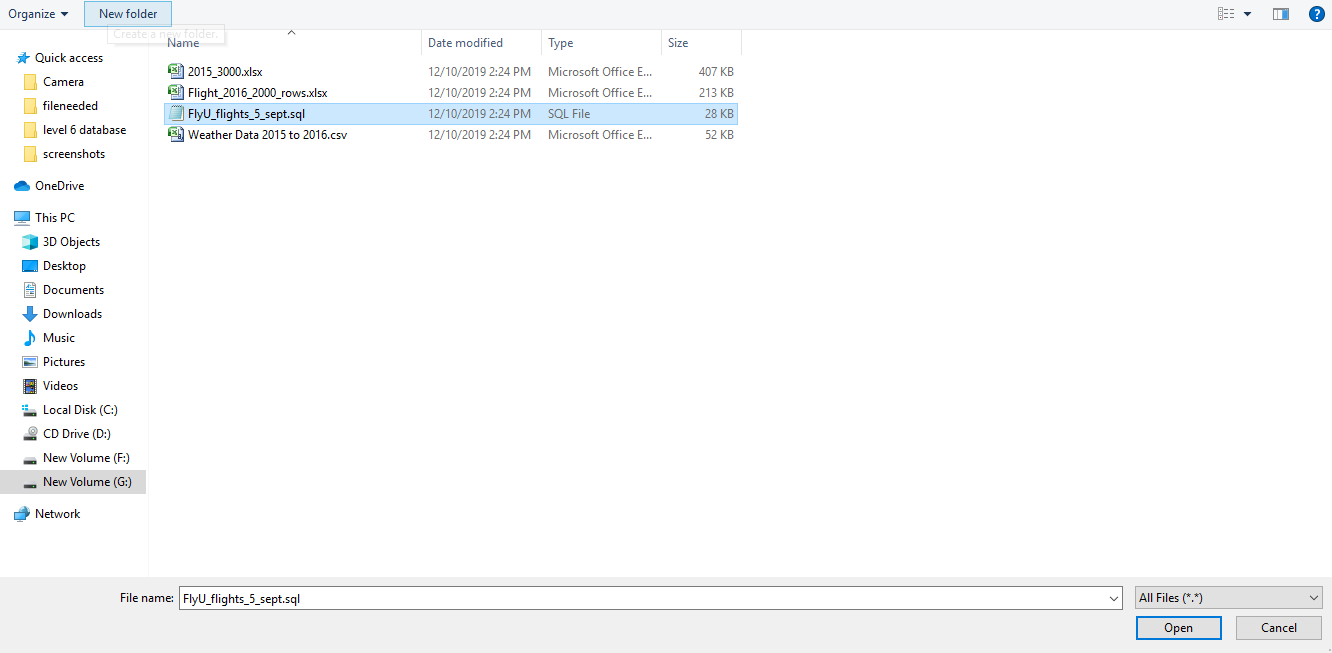
**The above shows that the table has been created**

****

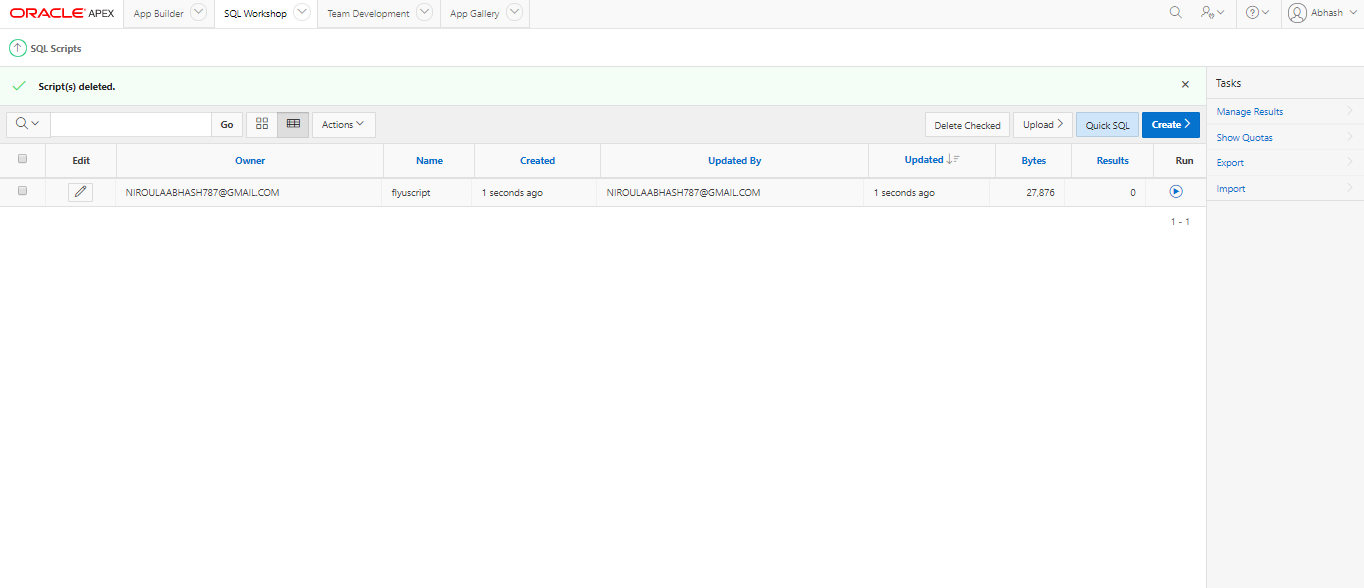
**Uploading script**

****

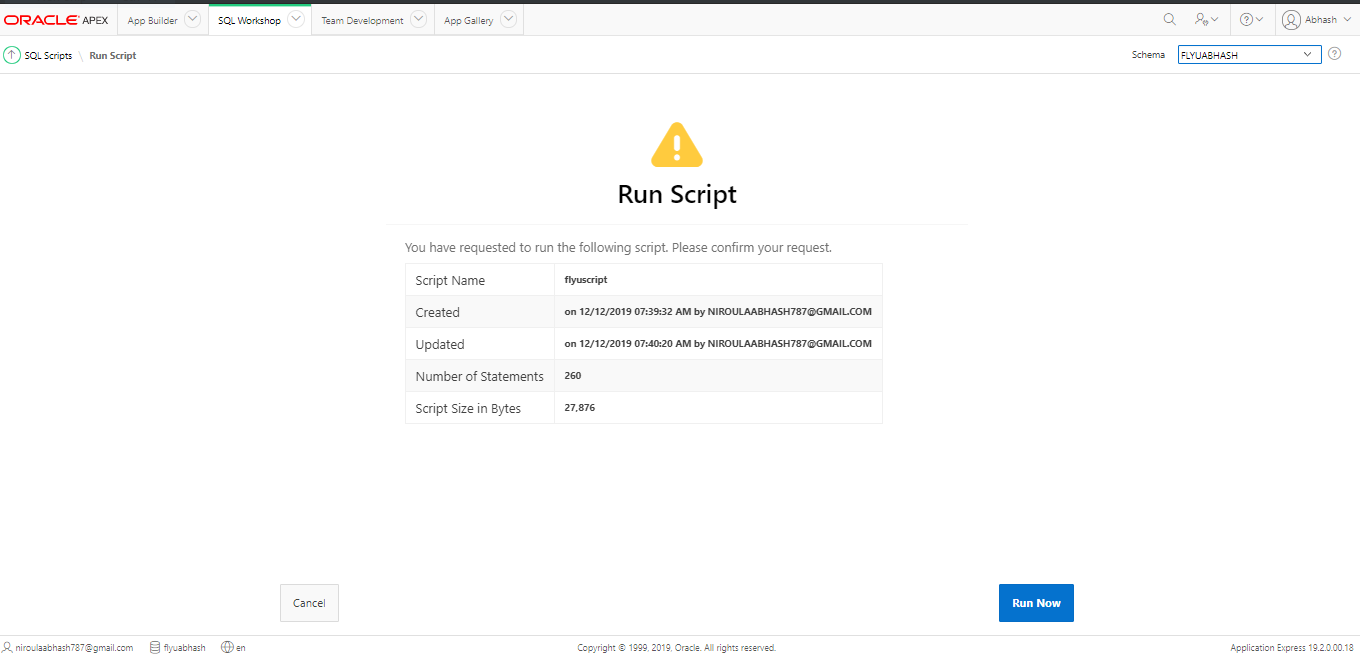
**Choosing file and giving script name**

****

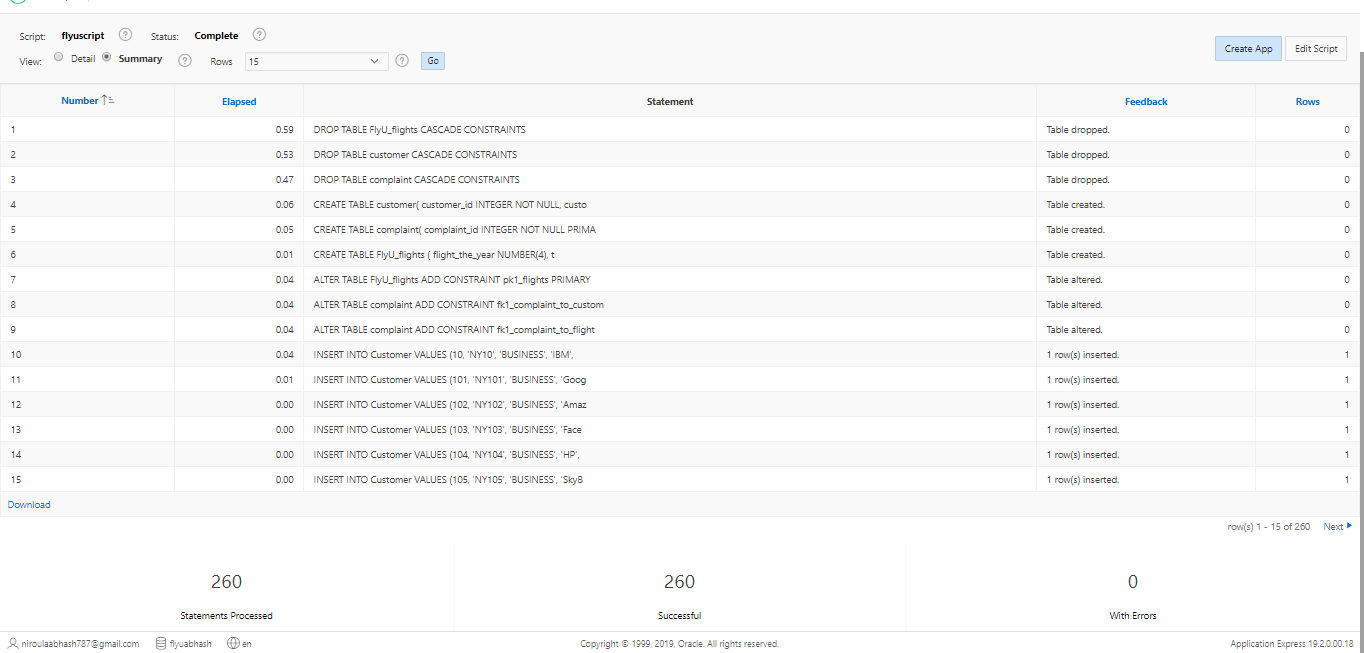
**Choosing file FlyU\_flights\_5\_sept.sql file**

****

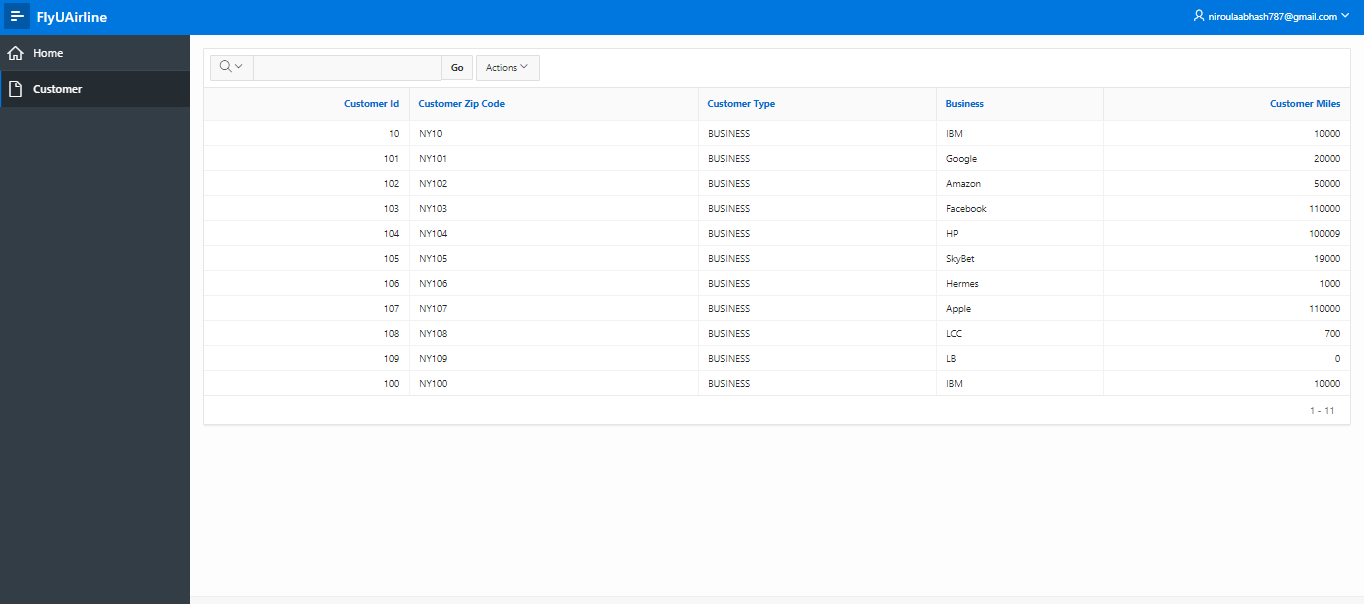
**Script has been uploaded**

****

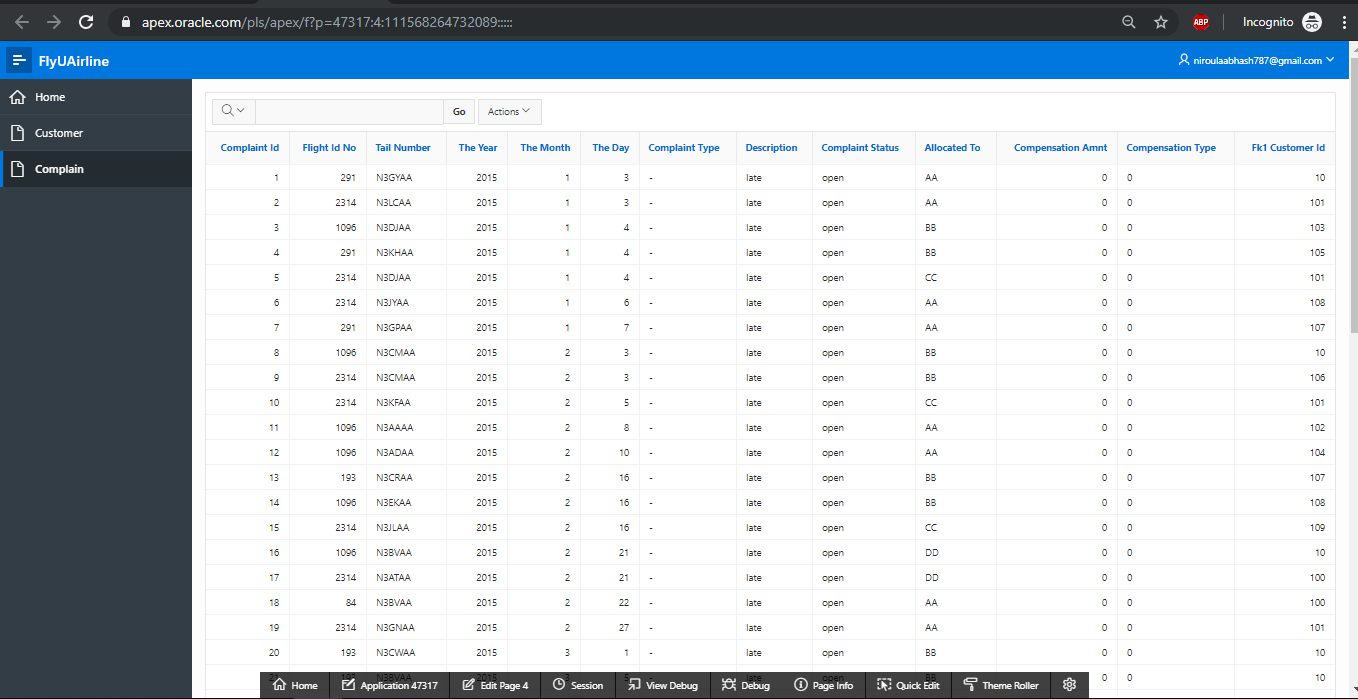
**Running the script**

****

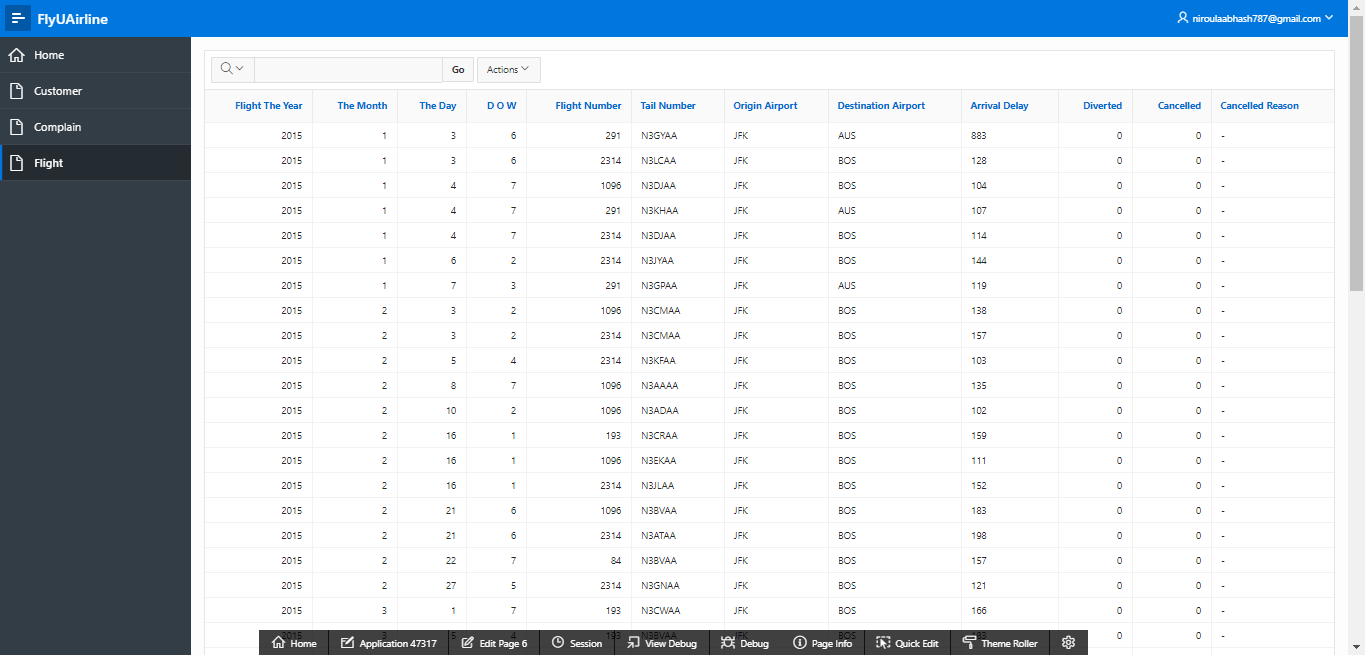
**The script was successfully loaded**

****

**Interactive report of customer table**

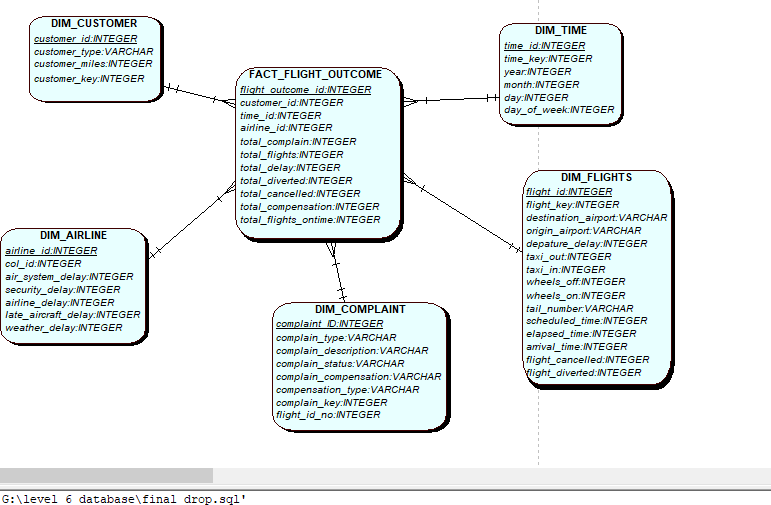
****

**Interactive report of complain table**

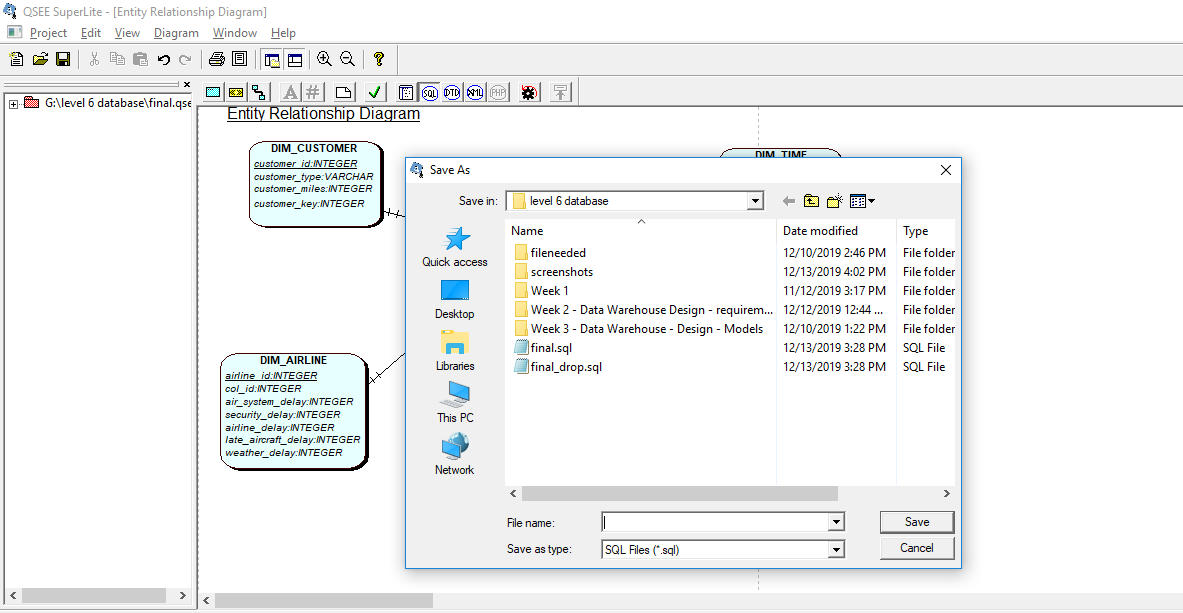
****

**Interactive report of Flight table**

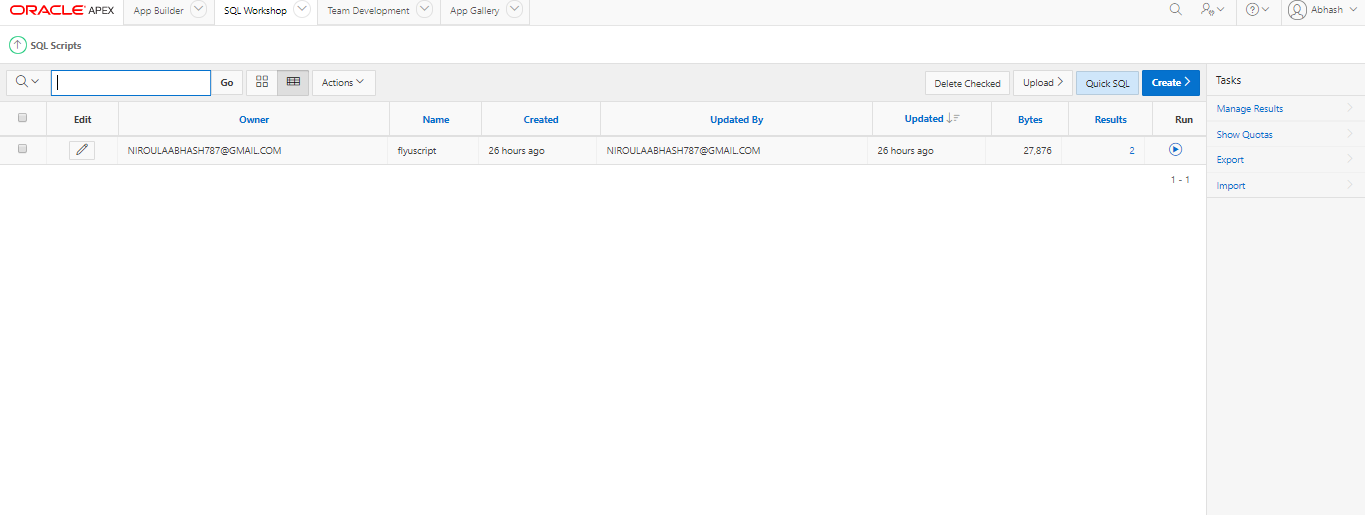
**Star Schema set up (DM environment)**

****

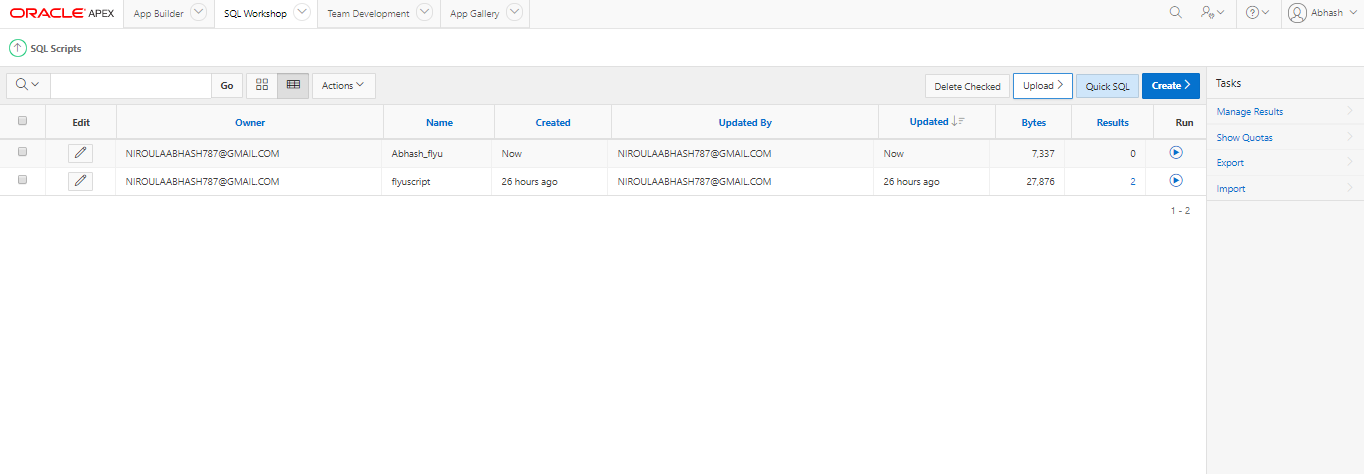
**Star Schema design in q see**

****

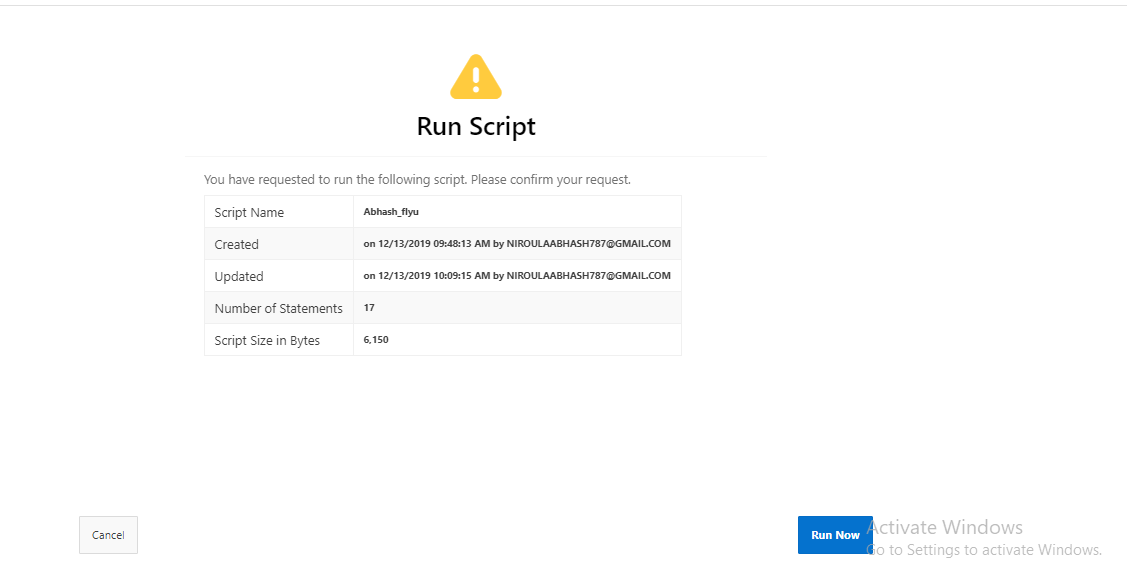
**Generating sql script using q see**

****

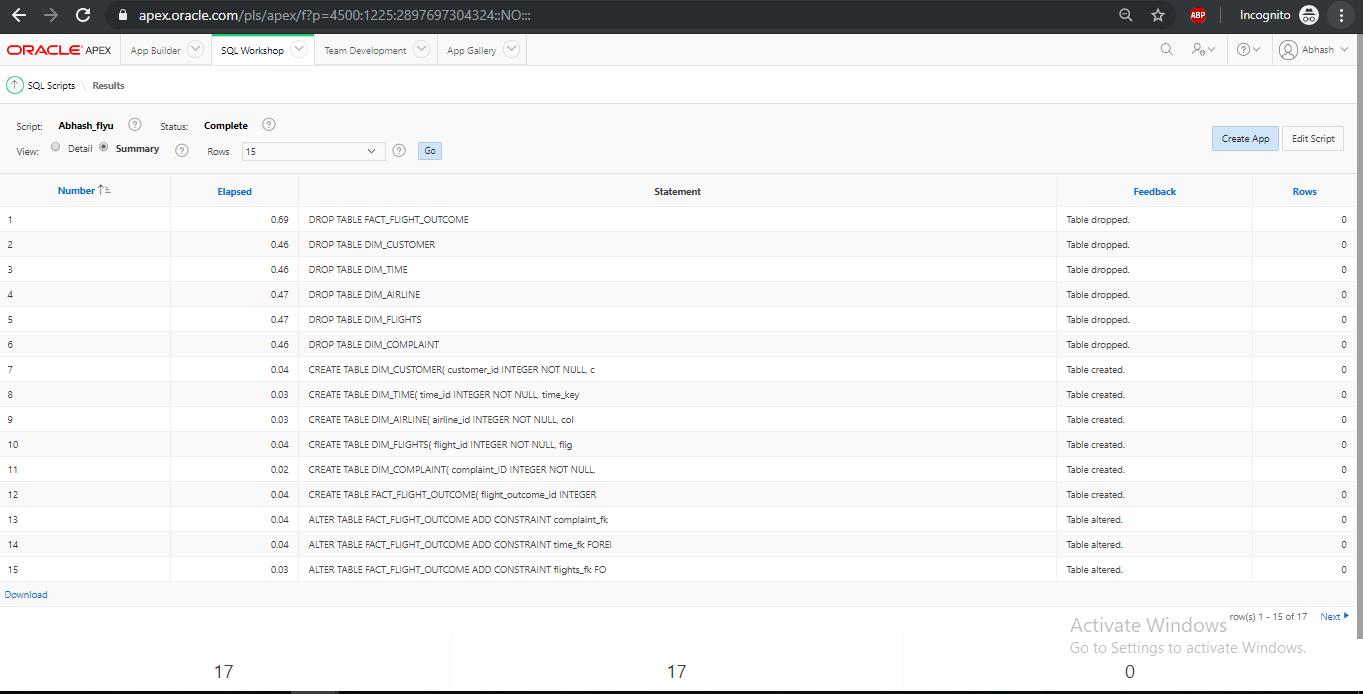
**Uploading the script in apex oracle**

****

**Script has been uploaded and ready to run**

****

**Running the script**

****

**Script has been successfully compiled**

**References**

Bose, B. (2018, 6 30). *Data science.* Retrieved from Digital Vidya: <https://www.digitalvidya.com/blog/introduction-to-data-warehousing/>

Franco, T. (2019, 3 7). *Blog.* Retrieved from Imaginary Cloud: <https://www.imaginarycloud.com/blog/oltp-vs-olap/>

Ramesh Sharda, D. D. (2014). *Business Intelligence and Analytics.* Pearson Education Limited .

Staff, H. ( 2019). Retrieved from Herzing Edu: <https://www.herzing.edu/blog/what-data-warehousing-and-why-it-important>

Tech Differences. (2016, 11 15). Retrieved from Tech Differences: <https://techdifferences.com/difference-between-oltp-and-olap.html>

Thomas M. Connolly, C. E. (2005). *Database Systems.* Pearson Education, Ltd.

University, s. f. (2019). *What is Data Warehousing and Why is it Important?* Retrieved from Herzing University: <https://www.herzing.edu/blog/what-data-warehousing-and-why-it-important>