Advanced Databases | Part 1 | Benjamin Wilson | C3444086

Table of Contents

[Task 1 | Design 2](#_Toc498958543)

[Star Schema and Identified Reports 2](#_Toc498958544)

[The 4th Report example 2](#_Toc498958545)

[Data Dictionary 3](#_Toc498958546)

[Task 2 | Loading the Data sources 4](#_Toc498958547)

[Data Source 1 LDS 4](#_Toc498958548)

[Data Source 2 SU 5](#_Toc498958549)

[Task 3 | ETL and Star Schema Loading Script 5](#_Toc498958550)

[Star Schema Implementation 5](#_Toc498958551)

[Staging Area 7](#_Toc498958552)

[Designing the Audit Log 8](#_Toc498958553)

[Load the data from the Leeds and Manchester placement tables. 11](#_Toc498958554)

[Check the audit log for issues 12](#_Toc498958555)

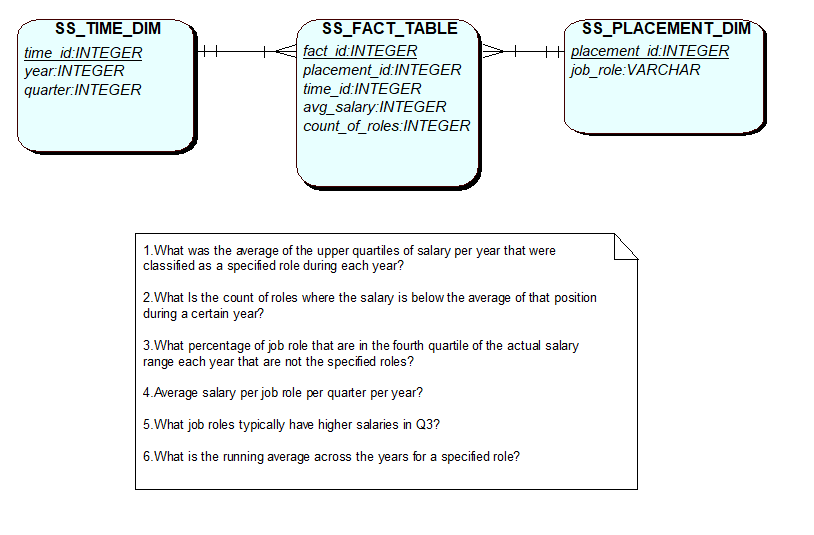
[Insert cleaned values into the final temporary table. 14](#_Toc498958556)

[Populate the Dimensions 15](#_Toc498958557)

[Populate the fact table 17](#_Toc498958558)

# Task 1 | Design

## Star Schema and Identified Reports



## The 4th Report example

Average salary per analyst type role per quarter per year?

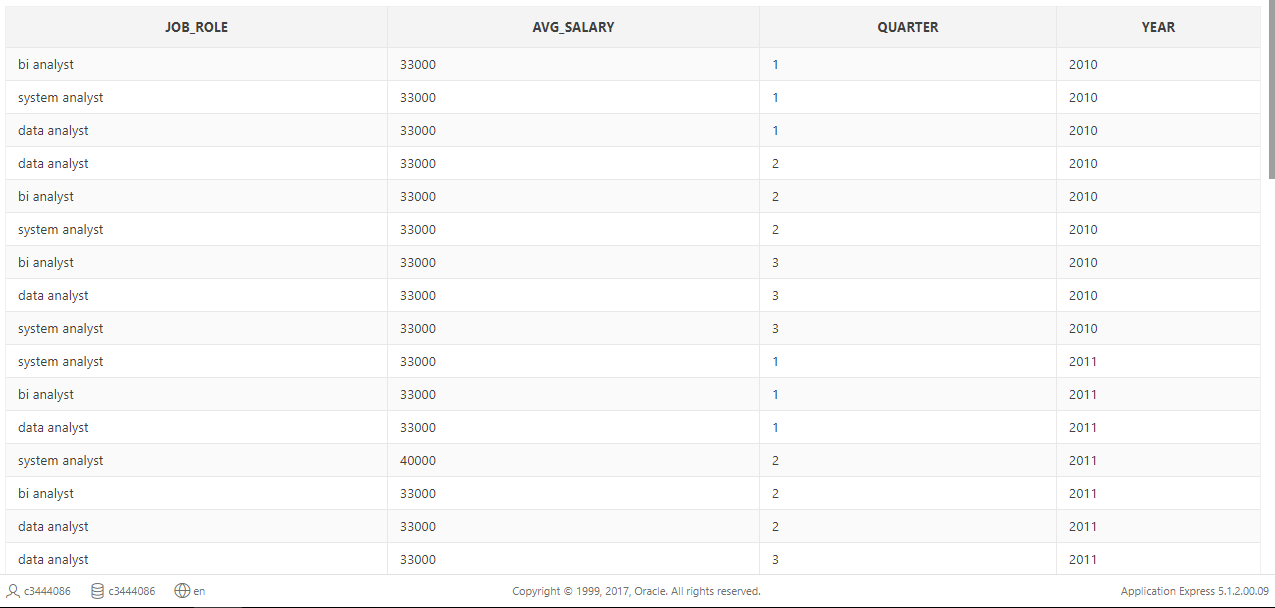
SELECT job\_role, avg\_salary,quarter,year FROM ss\_fact\_table

INNER JOIN ss\_time\_dim ON ss\_fact\_table.time\_id = ss\_time\_dim.time\_id

INNER JOIN ss\_job\_role\_dim ON ss\_fact\_table.job\_role\_id = ss\_job\_role\_dim.job\_role\_id

WHERE job\_role LIKE '%analyst%'

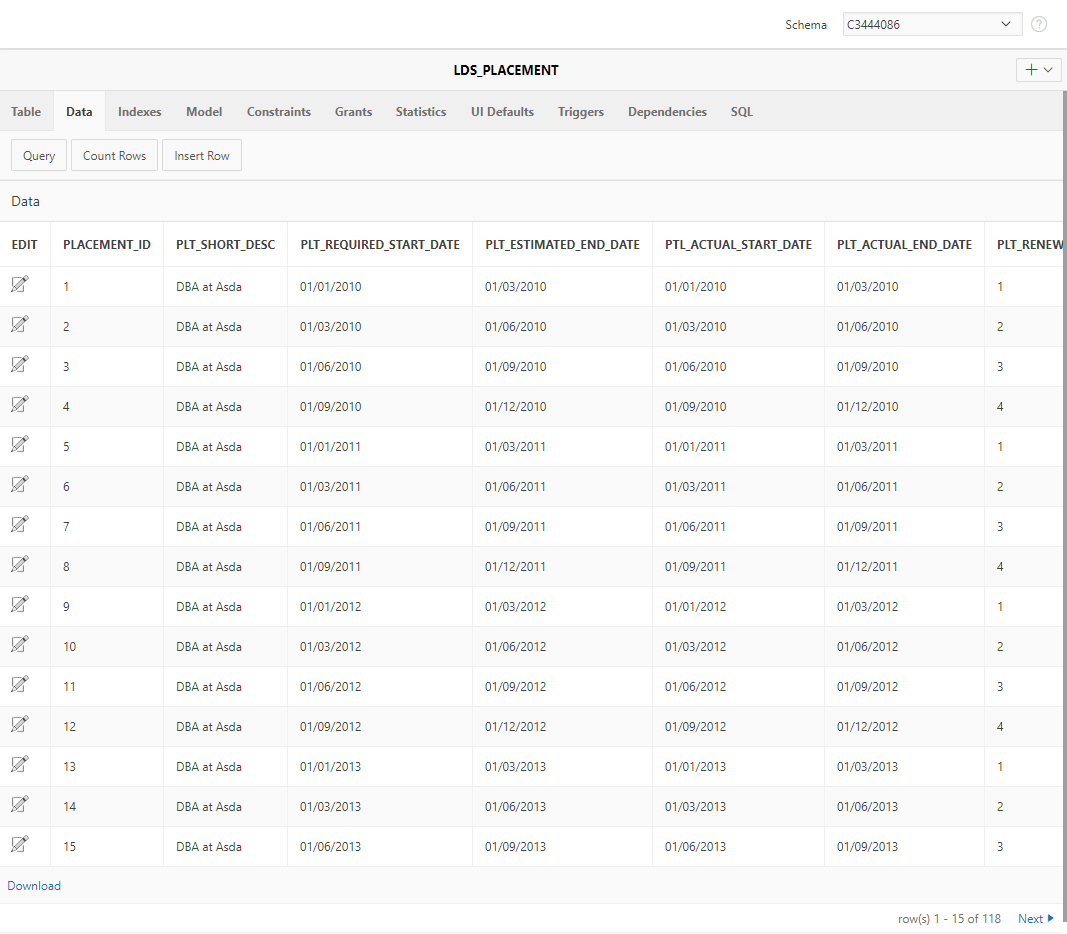
ORDER BY YEAR, QUARTER;



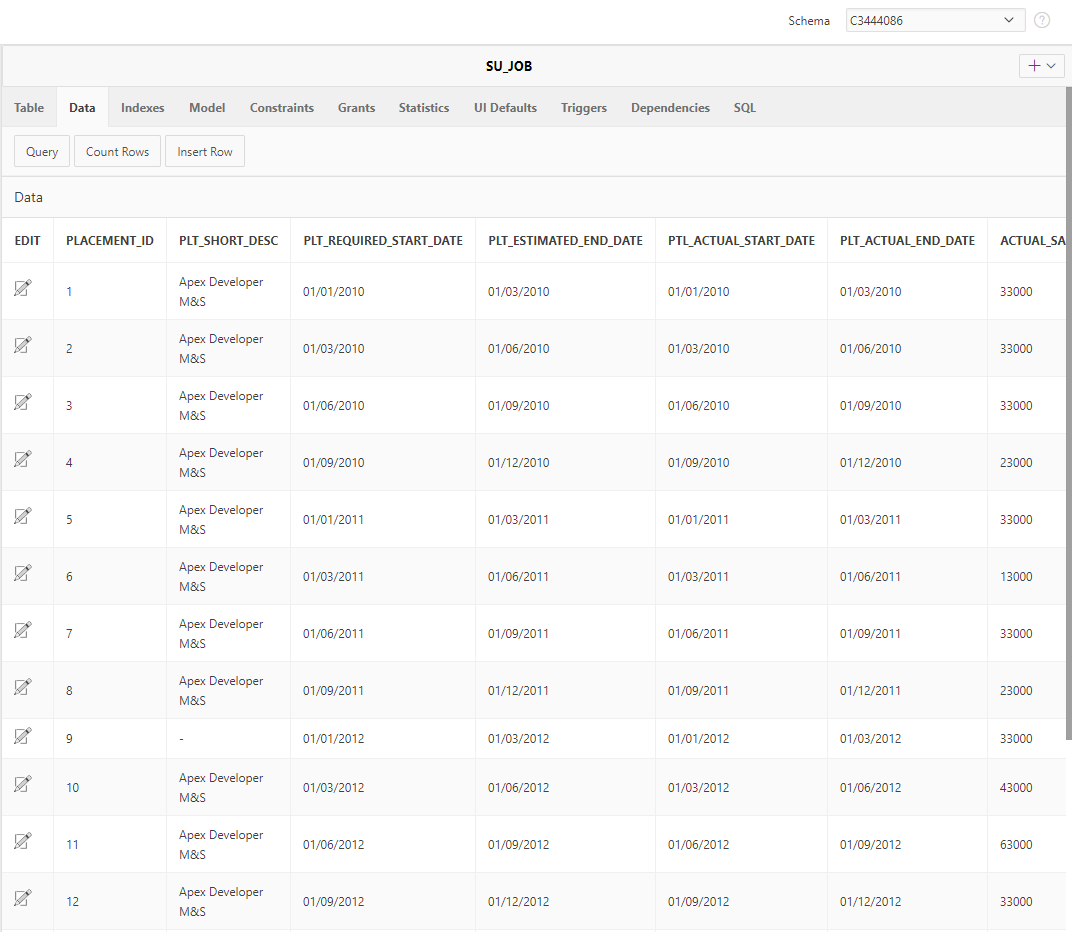
## Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Back Office | **Table Source(s)** | **Column** | **Type** | **Transformation(s)** | **Checks** | **Star Schema Destination** |
| placement\_id | lds\_placement  su\_jobs | placement\_id | VARCHAR2 | Concatenated a prefix to the id to avoid conflicts between the two sources. | UNIQUE, NOT NULL | Used for error checks in the audit log. |
| job\_role | lds\_placement  su\_job | plt\_job\_role  plt\_short\_desc | VARCHAR2 | Job descriptions don’t necessarily represent the ***actual*** job role in the LDS table. Conversions to the description/roles have been done for consistency. Converted all of them to lowercase. This will help with doing a LIKE comparison when reporting. | NOT NULL, LOWER CASE | ss\_job\_role\_dim.job\_role |
| placement\_start\_date | lds\_placement  su\_jobs | required\_start\_date | INTEGER | The date has also been split up into Grouped into years and then quarters. | NOT NULL,  CORRECT FORMAT | ss\_time\_dim.year  ss\_time\_dim.quarter |
| actual\_salary | lds\_placement  su\_jobs | actual\_salary | INTEGER |  | Checked for any abnormal outliers. | Used to generate measure. |

# Task 2 | Loading the Data sources

Data Source 1 LDS

## Data Source 2 SU



# Task 3 | ETL and Star Schema Loading Script

DROP TABLE SS\_FACT\_TABLE;

DROP TABLE SS\_JOB\_ROLE\_DIM;

DROP TABLE SS\_TIME\_DIM;

DROP TABLE ss\_temp\_unclean;

DROP TABLE ss\_temp\_clean;

DROP TABLE ss\_audit\_log;

## Star Schema Implementation

DROP SEQUENCE job\_role\_dim\_seq;

CREATE SEQUENCE job\_role\_dim\_seq START WITH 1 INCREMENT BY 1;

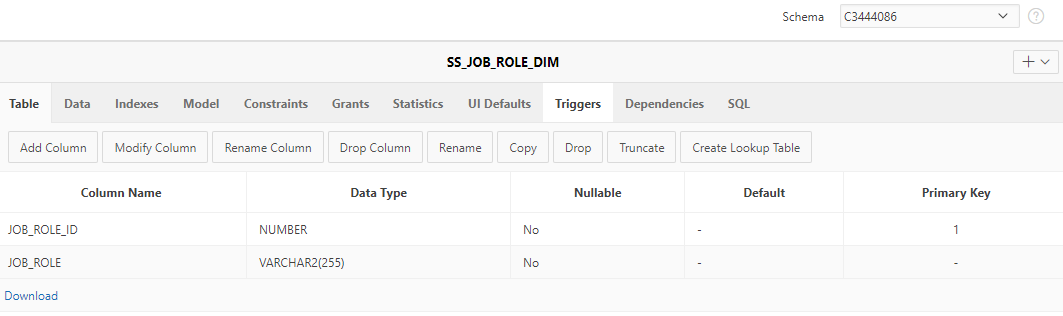
CREATE TABLE SS\_JOB\_ROLE\_DIM(

job\_role\_id INTEGER NOT NULL,

job\_role VARCHAR(255) NOT NULL,

CONSTRAINT pk\_SS\_PLACEMENT\_DIM PRIMARY KEY (job\_role\_id)

);



DROP SEQUENCE time\_seq;

CREATE SEQUENCE time\_seq START WITH 1 INCREMENT BY 1;

Dates type have been split up and cast into integers. This dimension supports more levels of granularity (quarter and year).

CREATE TABLE SS\_TIME\_DIM(

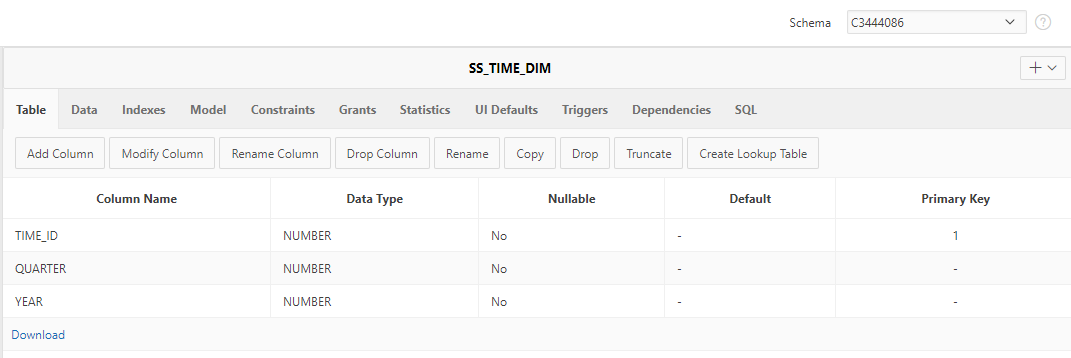
time\_id INTEGER NOT NULL,

quarter INTEGER NOT NULL,

year INTEGER NOT NULL,

CONSTRAINT pk\_SS\_TIME\_DIM PRIMARY KEY (time\_id)

);



DROP SEQUENCE fact\_table\_seq;

CREATE SEQUENCE fact\_table\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE SS\_FACT\_TABLE(

fact\_id INTEGER NOT NULL,

job\_role\_id INTEGER NOT NULL,

time\_id INTEGER NOT NULL,

avg\_salary INTEGER,

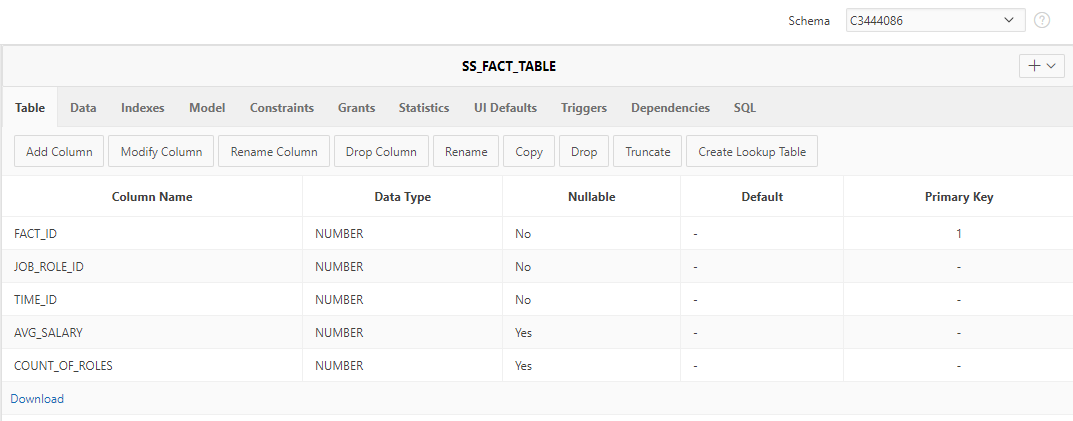
count\_of\_roles INTEGER,

CONSTRAINT pk\_fact\_id PRIMARY KEY (fact\_id),

CONSTRAINT fk\_placement\_dim FOREIGN KEY (job\_role\_id) REFERENCES SS\_JOB\_ROLE\_DIM(job\_role\_id),

CONSTRAINT fk\_time\_dim FOREIGN KEY (time\_id) REFERENCES SS\_TIME\_DIM(time\_id)

);



## Staging Area

Build a cleansing area where the data will first be loaded for checks quality checks. NOT NULL constraints have not been made here because we want the data in one place, regardless of quality, to simplify the error checking process by looking at a single table.

CREATE TABLE ss\_temp\_unclean (

placement\_id VARCHAR2(255),

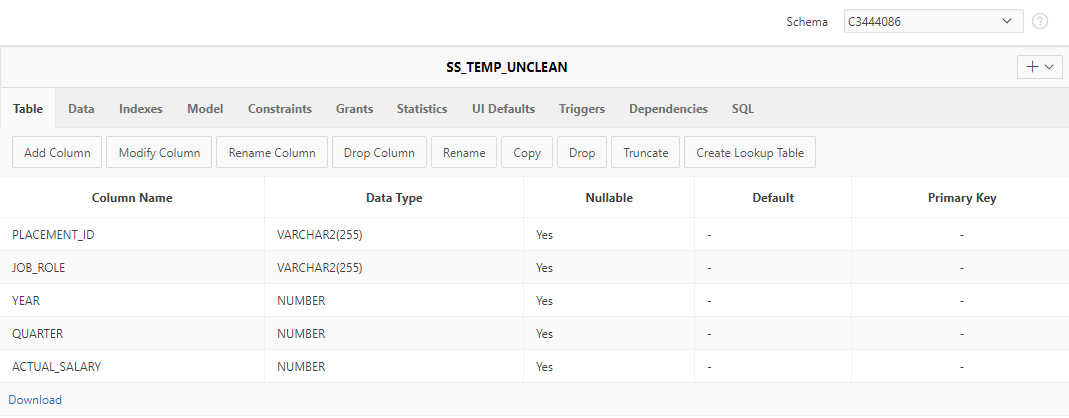
job\_role VARCHAR2(255),

year INTEGER,

quarter INTEGER,

actual\_salary INTEGER

);



Create a clean data table where checked and clean data will be placed before being put on the SS tables. Constraints now added.

CREATE TABLE ss\_temp\_clean (

placement\_id VARCHAR2(255) NOT NULL,

job\_role VARCHAR2(255) NOT NULL,

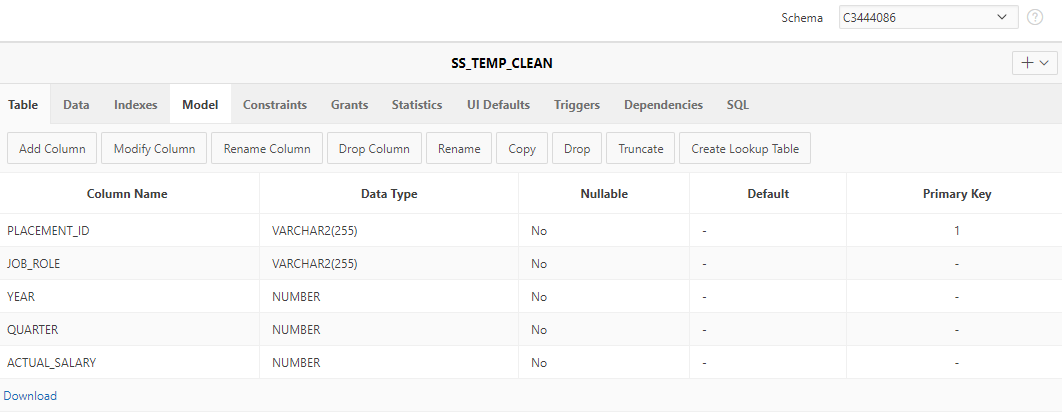
year INTEGER NOT NULL,

quarter INTEGER NOT NULL,

actual\_salary INTEGER NOT NULL,

CONSTRAINT pk\_placement\_id PRIMARY KEY ( placement\_id )

);



## Designing the Audit Log

The Audit log will log any potential issues with the data and add error codes to identify what type of data error it is for precision checking.

Error codes examples:

err:1 = Invalid Data

err:2 = IS NULL

err:3 = Abnormal Value

Audit logging will be triggered whenever an item is added to the ss\_temp\_unclean table.

DROP SEQUENCE audit\_seq;

CREATE SEQUENCE audit\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE ss\_audit\_log (

log\_id INTEGER NOT NULL,

id\_of\_occurrence VARCHAR2(255),

errcode VARCHAR2(8),

col VARCHAR2(255),

message VARCHAR2(255),

item\_in\_question VARCHAR2(255),

time\_stamp DATE,

checked VARCHAR2(16),

CONSTRAINT pk\_log\_id PRIMARY KEY ( log\_id )

);

CREATE OR REPLACE EDITIONABLE TRIGGER "AUDIT\_LOGGER" AFTER

INSERT ON ss\_temp\_unclean

BEGIN

Find any dates that are greater than the current date and log it.

INSERT INTO ss\_audit\_log

( SELECT

audit\_seq.NEXTVAL,

placement\_id,

'err:1',

'year',

'Invalid Date. Greater than current date!',

year,

SYSDATE,

'Unchecked'

FROM

ss\_temp\_unclean

WHERE

year > EXTRACT(YEAR FROM SYSDATE)

);

Check if Job role is null.

INSERT INTO ss\_audit\_log

( SELECT

audit\_seq.NEXTVAL,

placement\_id,

'err:2',

'job\_role',

'Is Null!',

'(null)',

SYSDATE,

'Unchecked'

FROM

ss\_temp\_unclean

WHERE

job\_role IS NULL

);

Find Potential Outliers that may be errors i.e. a value that is dispersed so abnormally far from the median. Can be reused if values change across time as it dynamically assesses the median.

INSERT INTO ss\_audit\_log

( SELECT

audit\_seq.NEXTVAL,

ss\_temp\_unclean.placement\_id,

'err:3',

'actual\_salary',

'Value abnormally far from the median!',

actual\_salary,

SYSDATE,

'Unchecked'

FROM

ss\_temp\_unclean

INNER JOIN (

SELECT

placement\_id,

abs(actual\_salary - med) AS dispersion\_from\_median

FROM

(

SELECT

MEDIAN(actual\_salary) AS med

FROM

ss\_temp\_unclean

),

ss\_temp\_unclean

) r ON r.placement\_id = ss\_temp\_unclean.placement\_id

WHERE

--Coefficient may need to be adjusted. Dependent on what the client boundaries are.

dispersion\_from\_median >= 10 \* (

SELECT

MEDIAN(actual\_salary) AS med

FROM

ss\_temp\_unclean

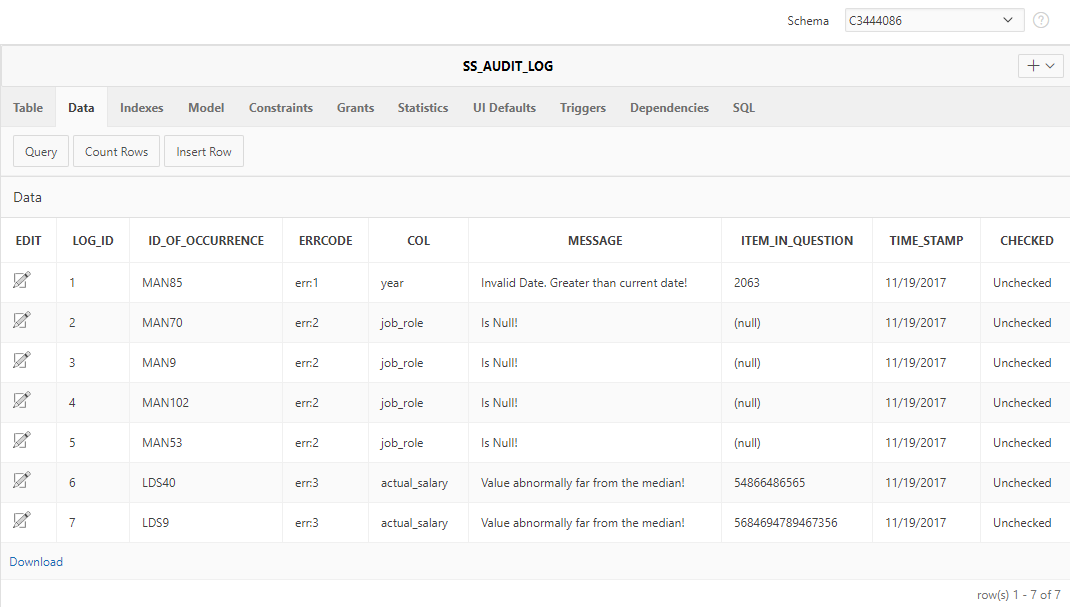
)

);

END;

/

ALTER TRIGGER "AUDIT\_LOGGER" ENABLE;



## Load the data from the Leeds and Manchester placement tables.

ID has been made unique by concatenating a prefix onto the original ID. This was done because else we would have matching instances of id coming from both tables. Done a join on the Leeds placement table because the actual job role was referenced by a foreign key to the job\_role table. Lowercase all job titles for easier LIKE comparisons.

INSERT INTO ss\_temp\_unclean

( SELECT

'LDS'

|| placement\_id,

lower(lds\_job\_role.job\_role\_desc),

EXTRACT(YEAR FROM plt\_required\_start\_date) AS year,

TO\_CHAR(TO\_DATE(plt\_required\_start\_date, 'DD/MM/YYYY'), 'Q') AS quarter,

actual\_salary

FROM

lds\_placement

INNER JOIN lds\_job\_role ON lds\_job\_role.job\_role\_id = lds\_placement.fk3\_job\_role\_id

UNION

SELECT

'MAN'

|| placement\_id,

lower(plt\_short\_desc),

EXTRACT(YEAR FROM plt\_required\_start\_date) AS year,

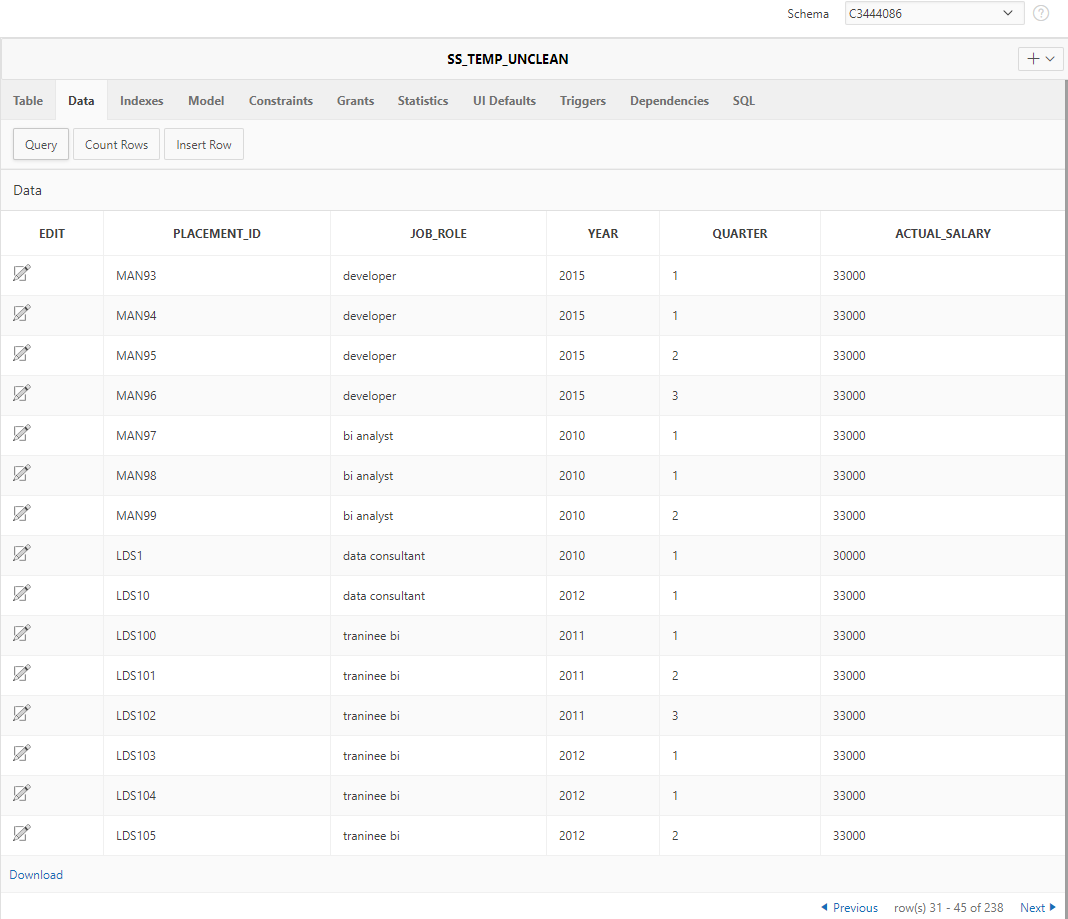
TO\_CHAR(TO\_DATE(plt\_required\_start\_date, 'DD/MM/YYYY'), 'Q') AS quarter,

actual\_salary

FROM

su\_job

);



## Check the audit log for issues

Make NULL job\_role values 'Unknown'. These will still be inserted into the star schema because report may not be dependent on known job\_role values, namely, a report could be that we want to analyse the unknown roles in some manner.

UPDATE ss\_temp\_unclean

SET

job\_role = 'Unknown'

WHERE placement\_id IN(

SELECT

placement\_id

FROM

ss\_audit\_log,

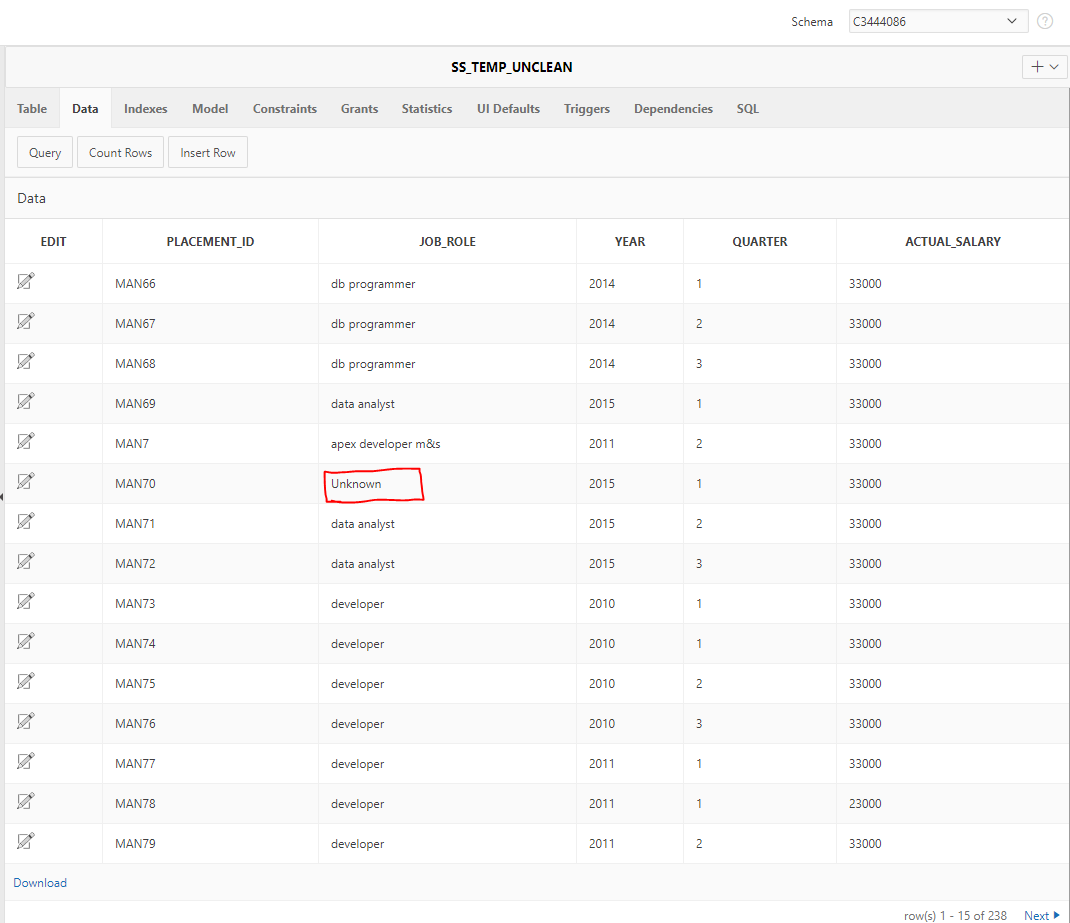
ss\_temp\_unclean

WHERE

ss\_temp\_unclean.placement\_id = ss\_audit\_log.id\_of\_occurrence

AND ss\_audit\_log.errcode LIKE '%err:2%'

);



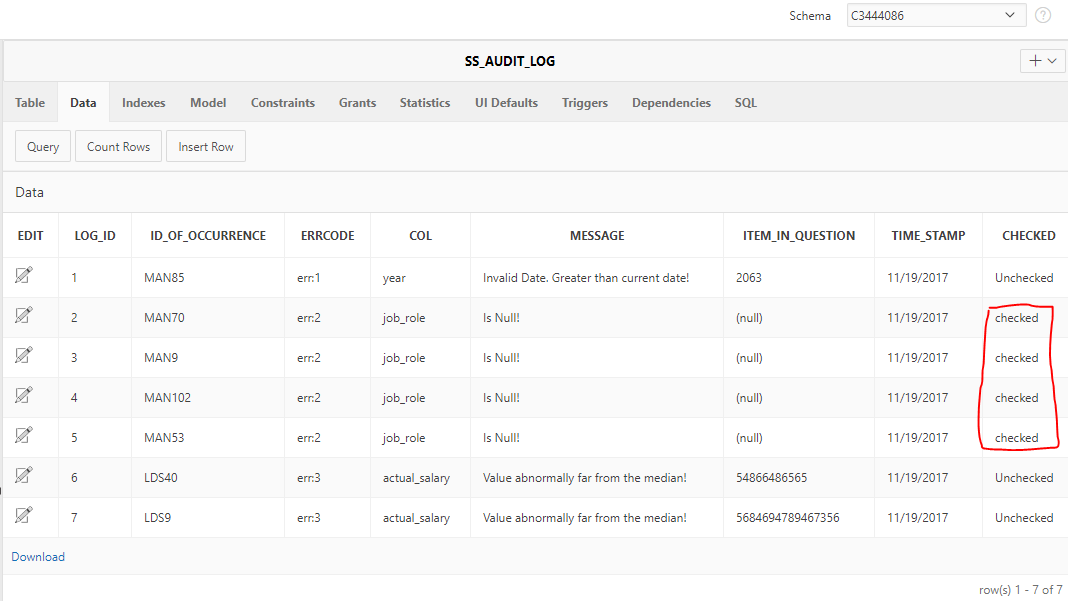
Update the audit to say that the value has been checked.

UPDATE ss\_audit\_log

SET

checked = 'checked'

WHERE col LIKE '%job\_role%' AND errcode LIKE '%err:2%';



## Insert cleaned values into the final temporary table.

Because of the nature of the reports that this data mart will support, I needed to completely remove any of the values because they will affect the statistical values needed to create an accurate report. Cannot replace the salaries with values that may, for example, affect the mean, max or min. Removing them will keep the reports accurate. The same goes for years and groupings on rolling averages.

To filter out the unwanted values, the audit contains a checked column the specifies whether it is okay to enter the values into the dimensions, this is referred to by a subquery.

INSERT INTO ss\_temp\_clean

( SELECT

placement\_id,

job\_role,

year,

quarter,

actual\_salary

FROM

ss\_temp\_unclean

WHERE

placement\_id NOT IN (

SELECT

id\_of\_occurrence

FROM

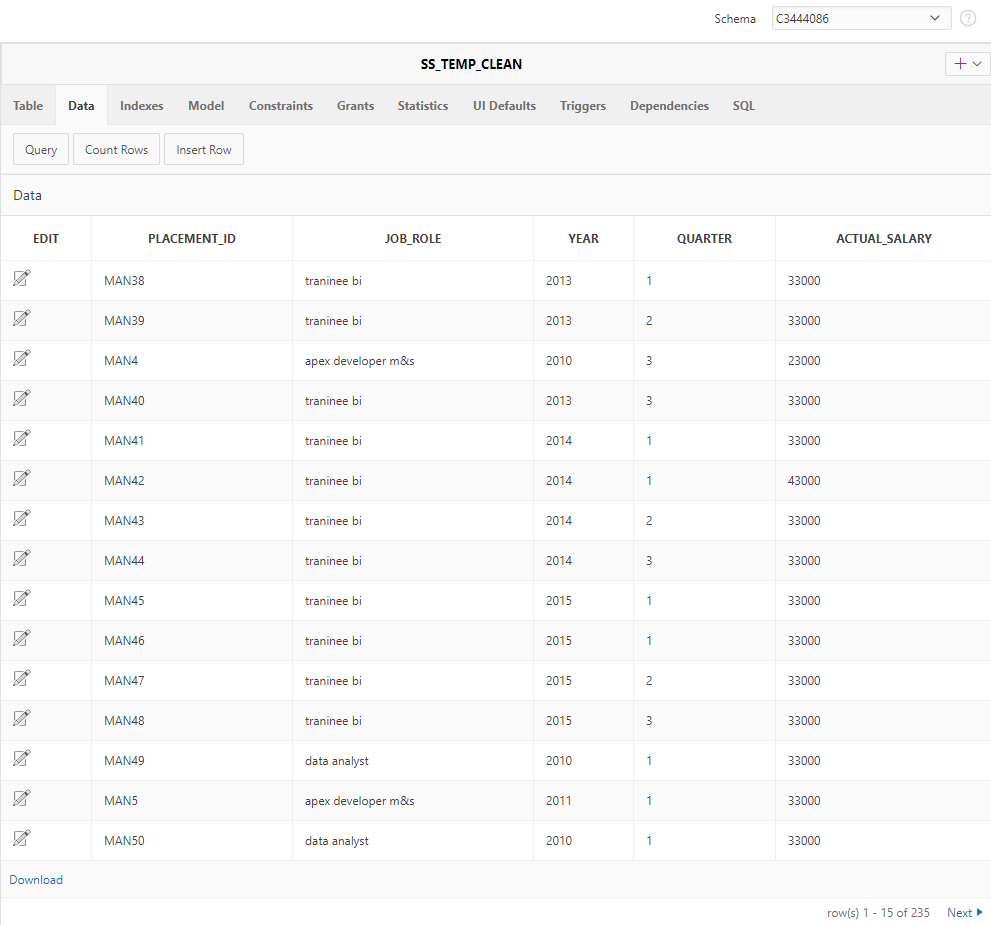
ss\_audit\_log

WHERE

ss\_audit\_log.checked LIKE '%Unchecked%'

)

);



## Populate the Dimensions

INSERT INTO ss\_time\_dim(

SELECT time\_seq.NEXTVAL, quarter, year FROM(

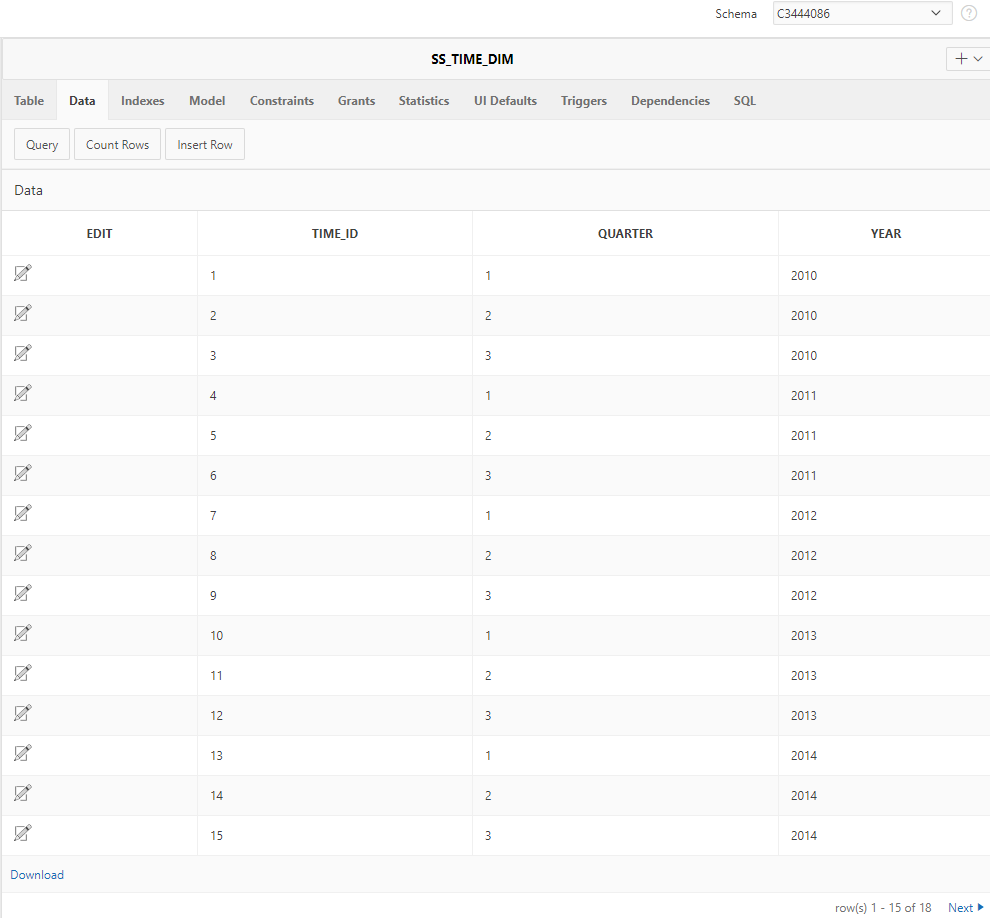
SELECT quarter, year, COUNT(\*) AS No\_Of\_Placements

FROM ss\_temp\_clean

GROUP BY quarter, year

ORDER BY year ASC)

);

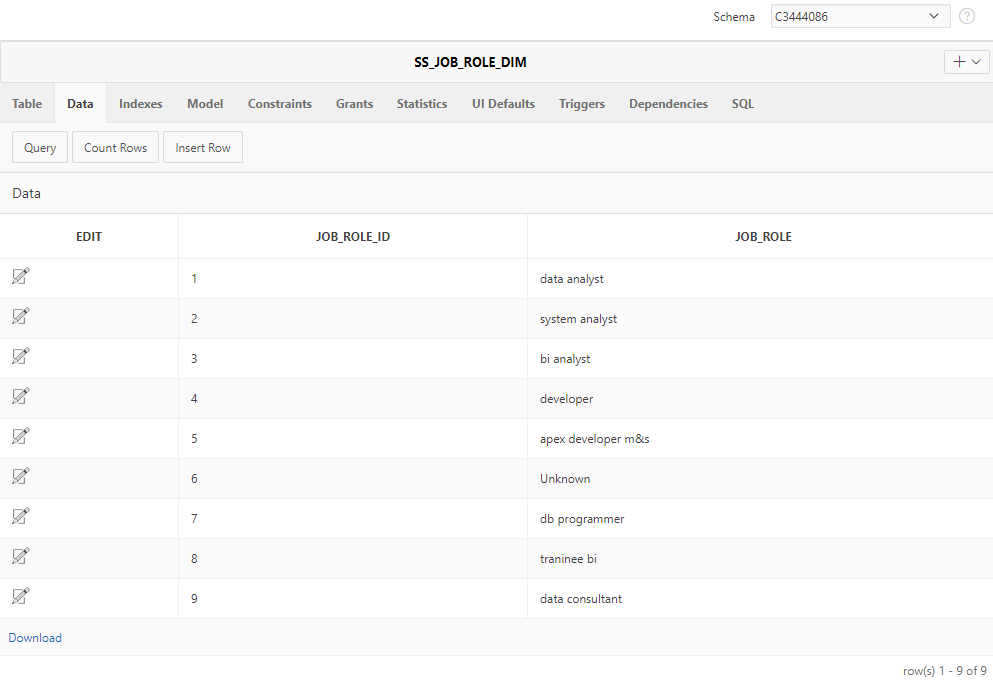


INSERT INTO ss\_job\_role\_dim(

SELECT job\_role\_dim\_seq.NEXTVAL ,job\_role FROM (

SELECT job\_role FROM ss\_temp\_clean GROUP BY job\_role)

);



## Populate the fact table

INSERT INTO ss\_fact\_table(

SELECT fact\_table\_seq.NEXTVAL, job\_role\_id, time\_id, Average\_Salary, Count\_of\_roles FROM (

SELECT ss\_job\_role\_dim.job\_role\_id, ss\_time\_dim.time\_id, ROUND(AVG(ss\_temp\_clean.actual\_salary),2) AS Average\_Salary ,COUNT(\*) AS Count\_of\_roles

FROM ss\_temp\_clean

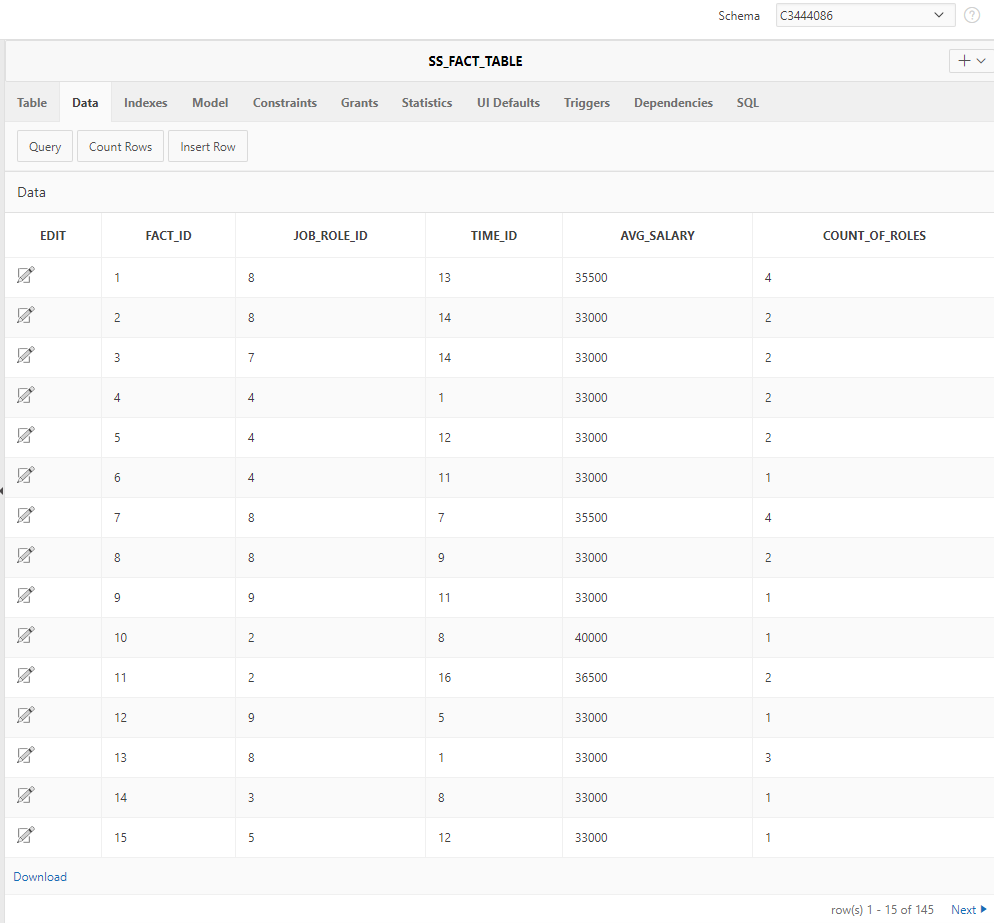
INNER JOIN ss\_time\_dim ON ss\_time\_dim.year = ss\_temp\_clean.year AND ss\_time\_dim.quarter = ss\_temp\_clean.quarter

INNER JOIN ss\_job\_role\_dim ON ss\_job\_role\_dim.job\_role = ss\_temp\_clean.job\_role

GROUP BY ss\_job\_role\_dim.job\_role\_id, ss\_time\_dim.time\_id

)

);



Script Results

