Database Term Project on Sales Transaction Application

By:

Karthik B Manjunath

Rohan U Sawant

(Claremont Graduate University)

Guided by:

Dr. Yan Li

December 2016

Table of Contents

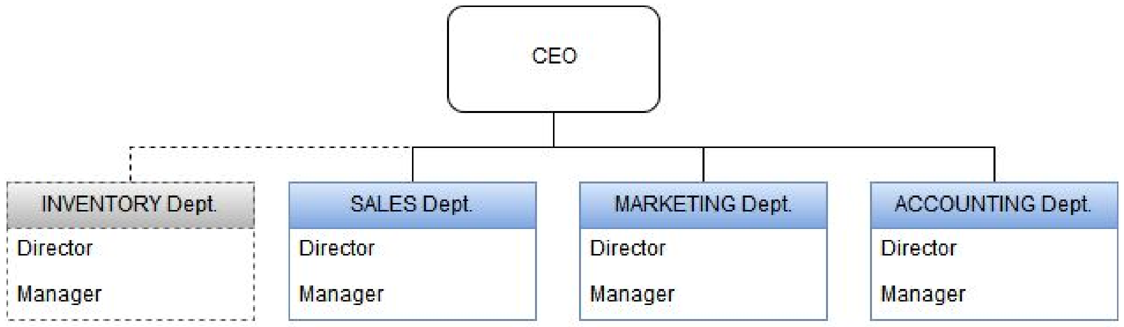
No table of contents entries found.

Table of Contents

Introduction

The purpose of this report is to design a database system for a sales transaction application in order to improve the company's current database system. This company sells 72 products, which are categorized into 5 categories and 22 subcategories. It has 5 different sales channels: direct sales, telesales, catalog, internet and partners, which are divided into 3 channel classes: direct, indirect, and others. 55,000 customers in 15 countries have purchased its products. 48,500 sales transactions have generated $4,368,743.93 in sales during the time period from 2012 to 2013, and 492,064 sales transactions have generated $44,230,567.8 in sales for 2014. There were 503 promotions between 2012 and 2014, with an average cost of $50,160.83 and a total cost of $25,230,900 for an average of 61.5 days. There were 9 categories and 22 subcategories of promotions.

As shown in Figure 1, the company currently has 4 main departments: CEO, Sales Department, Marketing Department, and Accounting Department; and 4 users: CEO, Sales Director, Marketing Director, and Accounting Director. Each department, including the CEO, has a specific interest in the sales transaction information. The CEO is a key stakeholder whose main responsibility is to have a vision for the company and lead their team to success. The CEO would like to see a broad overview of the company's performance.



The sales department will need information pertaining to revenue as a result of its sales activities. The sales department is responsible for setting sales goals, establishing training programs for the sales representatives, and advising the sales representatives on ways to improve their sales and performance. The sales department also oversees the regional and local sales managers and their staff.

The marketing department is in charge of the development and implementation of the brand strategy. They also oversee the implementation of marketing campaigns. Thus, the marketing department is concerned about sales changes attributed to promotions and cost of promotions.

The accounting department is responsible for handling financial matters, such as accounts payable and account receivable. Therefore, the accounting department will be interested in company profits and losses, which is defined as the difference between revenue and cost.

Currently, only 4 users will have access to their department's business views: the CEO, Sales Director, Marketing Director, and Accounting Director. Based on their needs, some of these individuals also have access to confidential customer information. In the future, we recommend creating a high-level business view for the Inventory Director, who is responsible for inventory, order fulfillment, and overseeing warehouse employees. This would include information such as the average time to ship. Further, 4 additional business views could be created for the manager in each department. These views would contain more detailed sales transaction information, but would exclude confidential customer information. These additional business views would allow the manager to support the director in day-to-day operations.

Data Understanding

Our team found that there were some inconsistencies in the data as follows:

1. In the LI_PRODUCTS table, there were 6 product categories and 72 products. The “Electronics” category was inconsistent, as it had two versions “Electronics” and “ELECTRONICS”.
2. In the LI_PROMOTIONS table, there were 503 distinct promotions. There was one mistake in this table. Two promotions had “ad news” as the promotion category, but “ad news” is supposed to be the subcategory.
3. In the LI_CHANNELS table the CHANNEL_ID was not consistent because there was a foreign key reference in other tables with a NUMBER datatype.
4. In the LI_CUSTOMERS_EXT table the CUST_MARITAL_STATUS attribute had many inconsistencies as same information was expressed in different forms.
5. In the LI_SALES_12_13 and LI_SALES_14 tables all the primary and foreign key constraints are defined as the VARCHAR datatype.

We identified the following columns contained useless data:

1. In the LI_CHANNELS table the column CHANNEL_TOTAL is unnecessary since all of the rows contain the same value “CHANNEL TOTAL”.
2. In the LI_CUSTOMER_INTX table the columns CUST_TOTAL and COUNTRY_TOTAL are unnecessary since the same value is listed in each column, “CUST_TOTAL” and “COUNTRY_TOTAL”, respectively.

3. In the LI_PRODUCTS table the columns PROD_STATUS and PROD_TOTAL are unnecessary since the same value is listed in each column, "PROD_STATUS" and "PROD_TOTAL", respectively.
4. In the LI_PROMOTIONS table the column PROMO_TOTAL is unnecessary because the rows contain the same value, "PROMO TOTAL".

The following table provides a summary of the database tables, including the total row count, distinct values, and cardinality.

***Table 1.** Database Tables Summary*

Table Name	Column Name	Total Row Count	Distinct Values	Cardinality
LI_CUSTOMERS_EXT	CUST_GENDER	55500	2	0.000036
LI_CUSTOMERS_INTX	CUST_GENDER	55500	2	0.000036
LI_CUSTOMERS_INTX	COUNTRY_REGION	55500	5	0.0000900
LI_CUSTOMERS_EXT	CUST_CREDIT_LIMIT	55500	8	0.0001441
LI_CUSTOMERS_INTX	COUNTRY_SUBREGION	55500	10	0.000180
LI_CUSTOMERS_EXT	CUST_MARITAL_STATUS	55500	12	0.000216
LI_CUSTOMERS_EXT	CUST_INCOME_LEVEL	55500	13	0.000234
LI_CUSTOMERS_INTX	COUNTRY_NAME	55500	19	0.000342
LI_SALES_14	QUANTITY_SOLD	492064	185	0.0003759
LI_CUSTOMERS_EXT	CUST_YEAR_OF_BIRTH	55500	75	0.00135
LI_SALES_14	SALE_DATE	492064	730	0.0014835
LI_SALES_14	SHIPPING_DATE	492064	734	0.001492
LI_SALES_14	PAYMENT_DATE	492064	738	0.0014998
LI_SALES_12_13	QUANTITY_SOLD	48500	116	0.00239
LI_CUSTOMERS_INTX	CUST_STATE_PROVINCE	55500	145	0.00261
LI_CUSTOMERS_INTX	CUST_CITY	55500	620	0.01117
LI_CUSTOMERS_INTX	CUST_POSTAL_CODE	55500	623	0.011225
LI_PRODUCTS	PROD_UNIT_OF_MEASURE	72	1(U)	0.01388
LI_PRODUCTS	PROD_PACK_SIZE	72	1(P)	0.01388
LI_PRODUCTS	SUPPLIER_ID	72	1	0.01388
LI_SALES_12_13	SALE_DATE	48500	728	0.0150

LI_SALES_12_13	SHIPPING_DATE	48500	733	0.0151
LI_SALES_12_13	PAYMENT_DATE	48500	735	0.0151
LI_CUSTOMERS_INTX	CUST_LAST_NAME	55500	908	0.0163
LI_PROMOTIONS	PROMO_CATEGORY	503	9	0.0179
LI_CUSTOMERS_INTX	CUST_FIRST_NAME	55500	1300	0.0234
LI_PRODUCTS	PROD_WEIGHT_CLASS	72	2	0.0278
LI_CUSTOMERS_INTX	CUST_EMAIL	55500	1699	0.0306
LI_PROMOTIONS	PROMO_SUBCATEGORY	503	22	0.0437
LI_SALES_14	UNIT_PRICE	492064	25535	0.0519
LI_SALES_14	AMOUNT_SOLD	492064	33924	0.0689
LI_PRODUCTS	PROD_CAT_DESC	72	5	0.06944
LI_PRODUCTS	PROD_CATEGORY	72	6	0.0833
LI_SALES_12_13	UNIT_PRICE	48500	8439	0.174
LI_SALES_12_13	AMOUNT_SOLD	48500	10841	0.2235
LI_PRODUCTS	PROD_SUBCAT_DESC	72	21	0.291667
LI_PRODUCTS	PROD_SUBCATEGORY	72	21	0.291667
LI_PROMOTIONS	PROMO_END_DATE	503	190	0.3777
LI_PROMOTIONS	PROMO_BEGIN_DATE	503	192	0.38171
LI_PRODUCTS	PROD_MIN_PRICE	72	42	0.5833
LI_PRODUCTS	PROD_LIST_PRICE	72	42	0.5833
LI_CHANNELS	CHANNEL_CLASS	5	3	0.6
LI_CUSTOMERS_INTX	CUST_STREET_ADDRESS	55500	50945	0.91792
LI_CUSTOMERS_INTX	CUST_MAIN_PHONE_NUMBER	55500	51000	0.9189
LI_PRODUCTS	PROD_NAME	72	71	0.98611
LI_PRODUCTS	PROD_DESC	72	71	0.98611
LI_PROMOTIONS	PROMO_COST	503	501	0.99602
LI_CHANNELS	CHANNEL_DESC	5	5	1
LI_PROMOTIONS	PROMO_NAME	503	503	1

The above table provides the cardinality of all the attributes in all the tables:

1. **LI_CHANNELS -** CHANNEL_CLASS(2 DISTINCT VALUES OUT OF 5)
2. **LI_CUSTOMERS_INTX -** CUST_GENDER(2 distinct values for 55000 rows), COUNTRY_REGION(5 distinct values for 55000 rows), COUNTRY_SUBREGION(10 distinct values for 55000 rows), COUNTRY_NAME(19 distinct values for 55000 rows),

CUST_STATE_PROVINCE(145 distinct values for 55000 rows), CUST_CITY(620 distinct values for 55000 rows), CUST_POSTAL_CODE(623 distinct values for 55000 rows)

3. **LI_CUSTOMERS_EXT -** CUST_CREDIT_LIMIT(8 distinct values for 55000 rows), CUST_MARITAL_STATUS(12 distinct values for 55000 rows), CUST_INCOME_LEVEL(13 distinct values for 55000 rows).
4. **LI_PRODUCTS -** PROD_UNIT_OF_MEASURE(1 distinct values for 72 rows), PROD_PACK_SIZE(1 distinct values for 72 rows)
5. **LI_PROMOTIONS -** PROMO_CATEGORY(9 distinct values for 503 rows), PROMO_SUBCATEGORY(22 distinct values for 503 rows),

The following attributes appear to be potential identifiers of embedded entities:

1. Product Category and Product Subcategory in LI_PRODUCTS
2. Customer and Country could be split in LI_CUSTOMERS_INTX as both are functionally independent
3. Promo Category and Promo Subcategory in LI_PROMOTIONS

The following attributes possess functional dependencies:

1. CHANNEL_CLASS depends on CHANNEL_ID, since CHANNEL_ID is the unique identifier
2. COUNTRY_NAME, COUNTRY_SUBREGION and COUNTRY_REGION depends on COUNTRY_ID, since COUNTRY_ID is the unique identifier

The following tables represent the same entity:

1. LI_SALES_12_13 and LI_SALES_14. Both tables have the same attribute, but are distinguished by the sale years. For example, LI_SALES_12_13 contains sales data for 2012 and 2013, while LI_SALES_14 contains sales data for 2014, 2015 and 2016. We chose to combine these two tables into one sales table for simplicity.

2. LI_CUSTOMERS_INTX and LI_CUSTOMERS_EXT. LI_CUSTOMERS_INTX contains data on customers obtained from internal sources, while LI_CUSTOMERS_EXT contains data on customers obtained from external courses. We also combined these two tables into one customer table for simplicity.

Data Modeling

Figure 2 displays the conceptual data model we designed based on our meetings with the client. Each SALES transaction has a PRODUCT, a CUSTOMER, a CHANNEL, and a PROMOTION. These are identified by the relevant IDs: PROD_ID, CUST_ID, CHANNEL_ID, and PROMO_ID. Each PRODUCT has a PRODUCT ID, PRODUCT SUBCATEGORY, and a SUPPLIER. A PRODUCT CATEGORY has one or more a PRODUCT SUBCATEGORY. A PRODUCT SUBCATEGORY has one or more PRODUCTS. Each PROMOTION has a PROMOTION ID and a PROMOTION NAME. Each CHANNEL has a CHANNEL ID, CHANNEL DESCRIPTION, and CHANNEL CLASS. Finally, each CUSTOMER has a CUSTOMER ID.

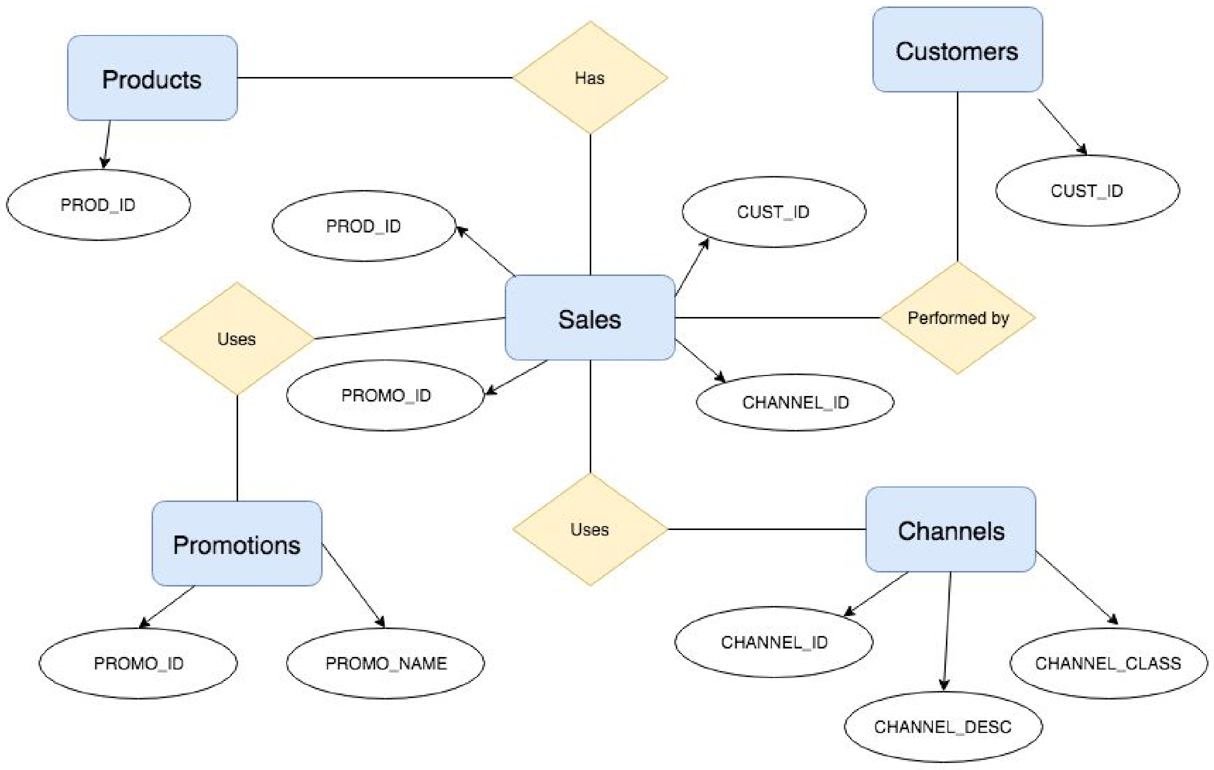


Figure 2*.Conceptual Data Model*

Application - Business Queries

Below are the business queries we designed for each user.

Sales Director

Sales_bvq1. In order to determine which month the company had the most revenue in a particular country, we provide a business query as follows:

```
CREATE OR REPLACE VIEW group3022.sales_bvq1 AS
SELECT count.country_name,
TO_CHAR(sale.sale_date, 'mm') AS month,
SUM(sale.amount_sold) AS revenue
```

FROM group3022.group3022_sales sale

INNER JOIN group3022.group3022_customers cust ON sale.cust_id = cust.cust_id

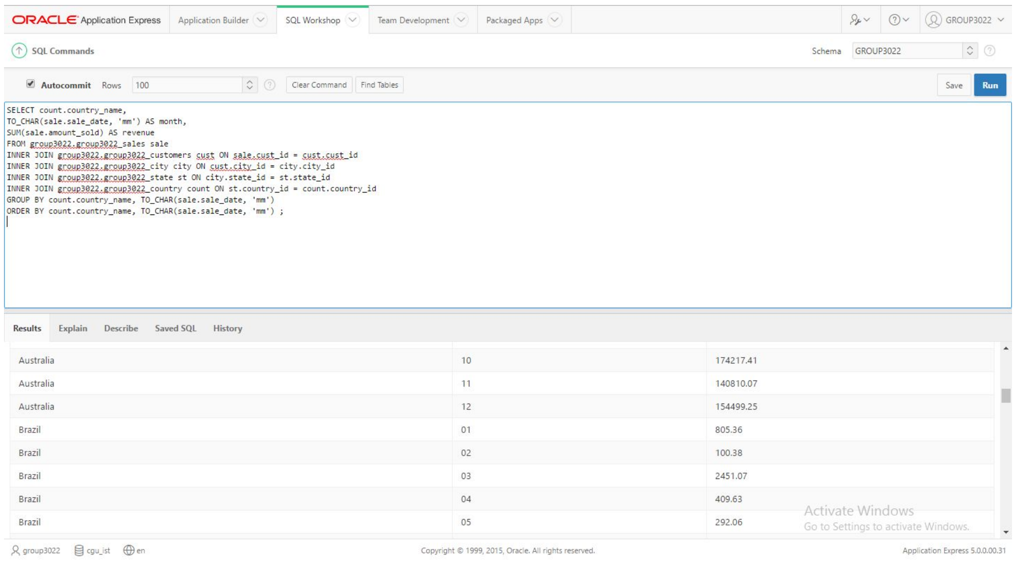
INNER JOIN group3022.group3022_city city ON cust.city_id = city.city_id

INNER JOIN group3022.group3022_state st ON city.state_id = st.state_id

INNER JOIN group3022.group3022_country count ON st.country_id = count.country_id

GROUP BY count.country_name, TO_CHAR(sale.sale_date, 'mm')

ORDER BY count.country_name, TO_CHAR(sale.sale_date, 'mm') ;



```
SELECT count.country_name,
TO_CHAR(sale.sale_date, 'mm') AS month,
SUM(sale.amount_sold) AS revenue
FROM group3022.group3022_sales sale
INNER JOIN group3022.group3022_customers cust ON sale.cust_id = cust.cust_id
INNER JOIN group3022.group3022_city city ON cust.city_id = city.city_id
INNER JOIN group3022.group3022_state st ON city.state_id = st.state_id
INNER JOIN group3022.group3022_country count ON st.country_id = count.country_id
GROUP BY count.country_name, TO_CHAR(sale.sale_date, 'mm')
ORDER BY count.country_name, TO_CHAR(sale.sale_date, 'mm') ;
```

Australia	10	174217.41
Australia	11	140810.07
Australia	12	154499.25
Brazil	01	805.36
Brazil	02	100.38
Brazil	03	2451.07
Brazil	04	409.63
Brazil	05	292.06

Sales_bvq2. In order to analyze which products had the highest sales in a particular subregion, we provide a business query as follows:

CREATE OR REPLACE VIEW group3022.sales_bvq2 AS

SELECT sr.country_subregion,

prod_name,

SUM(sale.amount_sold) AS revenue

FROM group3022.group3022_sales sale

INNER JOIN group3022.group3022_products prod ON sale.prod_id = prod.prod_id

INNER JOIN group3022.group3022_customers cust ON sale.cust_id = cust.cust_id

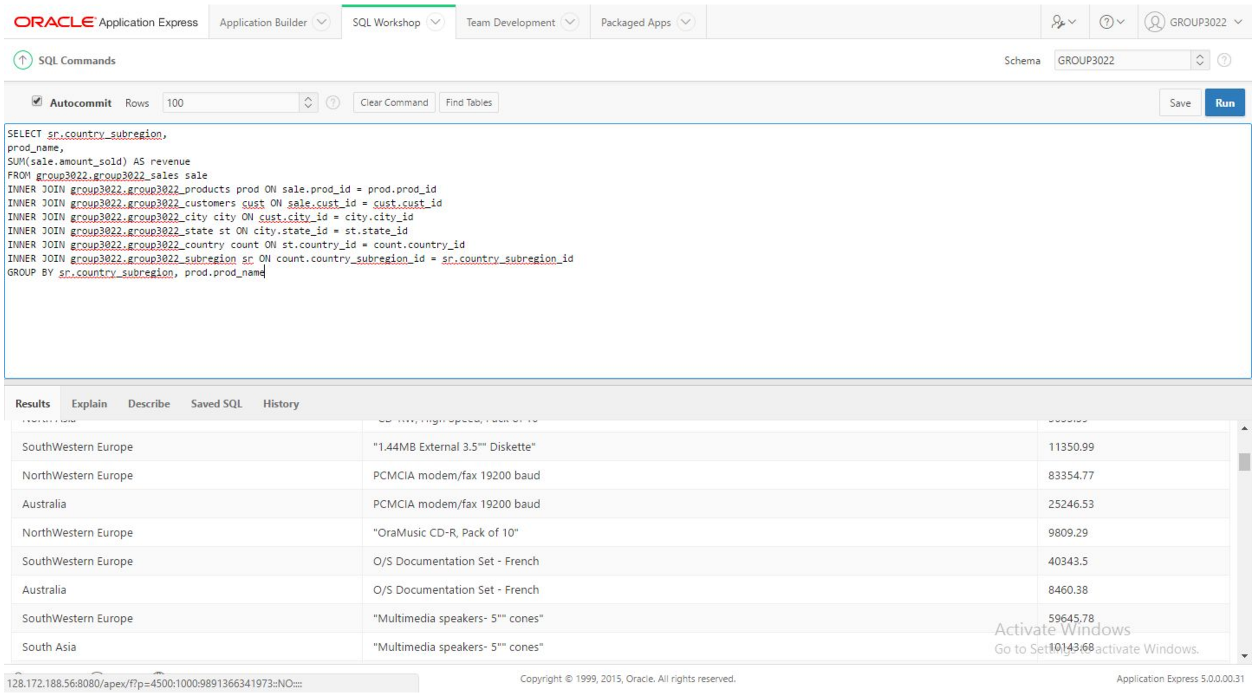
INNER JOIN group3022.group3022_city city ON cust.city_id = city.city_id

INNER JOIN group3022.group3022_state st ON city.state_id = st.state_id

INNER JOIN group3022.group3022_country count ON st.country_id = count.country_id

INNER JOIN group3022.group3022_subregion sr ON count.country_subregion_id = sr.country_subregion_id

GROUP BY sr.country_subregion, prod.prod_name



```
SELECT sr.country_subregion,
prod_name,
SUM(sale.amount_sold) AS revenue
FROM group3022.group3022_sales sale
INNER JOIN group3022.group3022_products prod ON sale.prod_id = prod.prod_id
INNER JOIN group3022.group3022_customers cust ON sale.cust_id = cust.cust_id
INNER JOIN group3022.group3022_city city ON cust.city_id = city.city_id
INNER JOIN group3022.group3022_state st ON city.state_id = st.state_id
INNER JOIN group3022.group3022_country count ON st.country_id = count.country_id
INNER JOIN group3022.group3022_subregion sr ON count.country_subregion_id = sr.country_subregion_id
GROUP BY sr.country_subregion, prod.prod_name
```

```
SouthWestern Europe   "1.44MB External 3.5"" Diskette"    11350.99
NorthWestern Europe   PCMCIA modem/fax 19200 baud         83354.77
Australia             PCMCIA modem/fax 19200 baud         25246.53
NorthWestern Europe   "OraMusic CD-R, Pack of 10"         9809.29
SouthWestern Europe   O/S Documentation Set - French      40343.5
Australia             O/S Documentation Set - French      8460.38
SouthWestern Europe   "Multimedia speakers- 5"" cones"    59645.78
South Asia            "Multimedia speakers- 5"" cones"    10143.68
```

128.172.188.56:8080/apex/f?p=4500:1000:9891366341973::NO:::

Copyright © 1999, 2015, Oracle. All rights reserved.

Application Express 5.0.0.00.31

Marketing Director

Market_bvq1. This business query is used to determine which product category had the highest revenue based on the promotion category.

CREATE OR REPLACE VIEW group3022.market_bvq1 AS

SELECT prod_category, promo_category, revenue

FROM (

SELECT prod_cat.prod_category, promo_cat.promo_category,

SUM(sale.amount_sold) AS revenue

FROM group3022.group3022_sales sale

INNER JOIN group3022.group3022_products prod

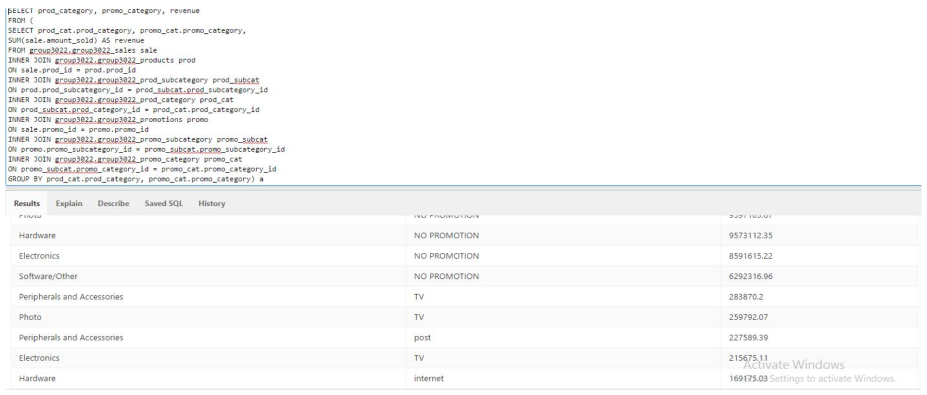
ON sale.prod_id = prod.prod_id

INNER JOIN group3022.group3022_prod_subcategory prod_subcat

ON prod.prod_subcategory_id = prod_subcat.prod_subcategory_id

INNER JOIN group3022.group3022_prod_category prod_cat

ON prod_subcat.prod_category_id = prod_cat.prod_category_id
INNER JOIN group3022.group3022_promotions promo
ON sale.promo_id = promo.promo_id
INNER JOIN group3022.group3022_promo_subcategory promo_subcat
ON promo.promo_subcategory_id = promo_subcat.promo_subcategory_id
INNER JOIN group3022.group3022_promo_category promo_cat
ON promo_subcat.promo_category_id = promo_cat.promo_category_id
GROUP BY prod_cat.prod_category, promo_cat.promo_category) a
ORDER BY revenue DESC;



```
SELECT prod_category, promo_category, revenue
FROM (
SELECT prod_cat.prod_category, promo_cat.promo_category,
SUM(sale.amount_sold) AS revenue
FROM group3022.group3022_sales sale
INNER JOIN group3022.group3022_products prod
ON sale.prod_id = prod.prod_id
INNER JOIN group3022.group3022_prod_subcategory prod_subcat
ON prod.prod_subcategory_id = prod_subcat.prod_subcategory_id
INNER JOIN group3022.group3022_prod_category prod_cat
ON prod_subcat.prod_category_id = prod_cat.prod_category_id
INNER JOIN group3022.group3022_promotions promo
ON sale.promo_id = promo.promo_id
INNER JOIN group3022.group3022_promo_subcategory promo_subcat
ON promo.promo_subcategory_id = promo_subcat.promo_subcategory_id
INNER JOIN group3022.group3022_promo_category promo_cat
ON promo_subcat.promo_category_id = promo_cat.promo_category_id
GROUP BY prod_cat.prod_category, promo_cat.promo_category) a
```

Results Explain Describe Saved SQL History

```
Hardware                      NO PROMOTION    9573112.35
Electronics                   NO PROMOTION    8591615.22
Software/Other                NO PROMOTION    6292316.96
Peripherals and Accessories   TV              283870.2
Photo                         TV              259792.07
Peripherals and Accessories   post            227589.39
Electronics                   TV              215675.11
Hardware                      internet        169175.03
```

Market_bvq3. This business query lists the customers in each state that are at the top 5% income level and whose percentage of distinct products purchased is less than 10% of the maximum number of distinct products purchased. This is helpful for determining which customers to target to increase sales within this customer segment.

CREATE OR REPLACE VIEW group3022.market_bvq3 AS
SELECT i.cust_state_province AS state, i.cust_first_name AS first_name, i.cust_last_name AS last_name, i.cust_income_level AS income_level,
i.distinct_prod_bought as "Unique # of Products Bought",
ROUND(i.percentage_bought_products, 2) AS "% of Comparison to Maximum #"
FROM (
SELECT h.cust_state_province, h.cust_first_name, h.cust_last_name, h.cust_income_level,
h.distinct_prod_bought, (h.distinct_prod_bought / h.max_distinct_prod_bought) * 100 AS percentage_bought_products
FROM (

```
SELECT g.cust_state_province, g.cust_first_name, g.cust_last_name, g.cust_income_level,
g.distinct_prod_bought, max(g.distinct_prod_bought) over (partition by g.cust_state_province)
AS max_distinct_prod_bought
FROM (
SELECT e.cust_state_province, e.cust_first_name, e.cust_last_name, e.cust_income_level,
count(distinct f.prod_id) AS distinct_prod_bought
FROM (
SELECT a.cust_id, a.cust_first_name, a.cust_last_name, a.cust_income_level,
c.cust_state_province, DENSE_RANK() OVER (PARTITION BY c.cust_state_province
ORDER BY a.cust_income_level DESC) AS rnum
FROM group3022.group3022_customers a
INNER JOIN group3022.group3022_city b ON a.city_id = b.city_id
INNER JOIN group3022.group3022_state c ON b.state_id = c.state_id
WHERE a.cust_income_level IS NOT NULL) e
INNER JOIN group3022.group3022_sales f ON e.cust_id = f.cust_id
WHERE e.rnum <= 5
GROUP BY e.cust_id, e.cust_first_name, e.cust_last_name, e.cust_income_level,
e.cust_state_province) g
) h
) i
WHERE i.percentage_bought_products < 10
ORDER BY i.cust_state_province, i.percentage_bought_products;
```

Accounting Director

Account_bv1. This business view lists the yearly revenue and the corresponding growth rate comparing the current year's revenue to the last year's revenue.

```
CREATE OR REPLACE VIEW group3022.account_bv1 AS
SELECT a.year, a.revenue,
ROUND(((a.revenue- LAG(a.revenue) OVER (ORDER BY a.year ASC)) / LAG(a.revenue)
OVER (ORDER BY a.year ASC)) * 100,2) AS growth_rate
FROM (
SELECT to_char(sale.payment_date,'YYYY') as year, sum(sale.amount_sold) as revenue
FROM group3022.group3022_sales sale
GROUP BY to_char(sale.payment_date,'YYYY')
ORDER BY to_char(sale.payment_date,'YYYY') ASC
) a;
```

CEO

CEO_bvq1. This is a materialized view for the CEO for the number of sales based on country.

```
CREATE MATERIALIZED VIEW group3022.ceo_bvq1
PCTFREE 10 PCTUSED 20 INITRANS 1 MAXTRANS 255
STORAGE ( INITIAL 8192 NEXT 8192 MINEXTENTS 1 PCTINCREASE 5)
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT country_name, count
FROM
(SELECT count.country_name, count(*) AS count
FROM group3022.group3022_sales sale
INNER JOIN group3022.group3022_products prod ON sale.prod_id = prod.prod_id
INNER JOIN group3022.group3022_customers cust ON sale.cust_id = cust.cust_id
INNER JOIN group3022.group3022_city city ON cust.city_id = city.city_id
INNER JOIN group3022.group3022_state st ON city.state_id = st.state_id
INNER JOIN group3022.group3022_country count ON st.country_id = count.country_id
GROUP BY count.country_name
) a
ORDER BY count desc;
```

CEO_bvq2. This is business view displays the top 3 selling products per year and the corresponding total revenue.

```
CREATE OR REPLACE VIEW group3022.ceo_bvq2 AS
SELECT e.year, e.position, e.prod_name, e.total_revenue
FROM (
SELECT d.year,
RANK() OVER (PARTITION BY d.year ORDER BY d.revenue DESC) AS position,
d.prod_name, revenue AS total_revenue
FROM (
SELECT c.year, c.prod_name, SUM(c.amount_sold) AS revenue
FROM (
SELECT EXTRACT(YEAR FROM b.sale_date) AS year, a.prod_name, b.amount_sold
FROM group3022.group3022_products a
INNER JOIN group3022.group3022_sales b ON a.prod_id = b.prod_id
) c
GROUP BY c.year,c.prod_name
) d
) e
WHERE position <= 3
ORDER BY e.year, e.position;
```

Security Requirements

Based on our query profile, Tables 2 and 3 display the security permission and prevention access, respectively. The CEO will have access to all views: Sales, Marketing, Accounting, and CEO. The Sales Director, Accounting Director, and Marketing Director will only have access to their respective business views. Providing access to only the relevant users maintains data integrity and confidentiality.

***Table 2**.Security Access Permission Matrix*

	Sales View	Marketing View	Accounting View	CEO View

CEO	yes	yes	yes	yes
Sales Director	yes	-	-	-
Accounting Director	-	-	yes	-
Marketing Director	-	yes	-	-

Table 3. *Security Access Prevention Matrix*

	Sales View	**Marketing View**	**Accounting View**	**CEO View**
CEO	-	-	-	-
Sales Director	-	yes	yes	yes
Accounting Director	yes	yes	-	yes
Marketing Director	yes	-	yes	yes

Relational Database Design

The entity-relationship diagram represented in Figure 3 is derived from the tables in "LIY26", which are normalized to 3NF in the "GROUP3022" schema. First, the unnecessary columns were removed and the embedded entities were grouped, as explained in the Data Understanding section and shown in figure 2. For example, we noticed that CUST_CITY, CUST_STATE, COUNTRY_NAME, CONTRY_SUBREGION, and COUNTRY_REGION in the original LI_CUSTOMERS_INTX table (CUSTOMERS in our schema), could be normalized to separate tables. The PRODUCTS and PROMOTIONS tables were also separated by PROD_CATEGORY and PROD_SUBCATEGORY. We defined the primary keys of these separate columns as "Column Name_ID".

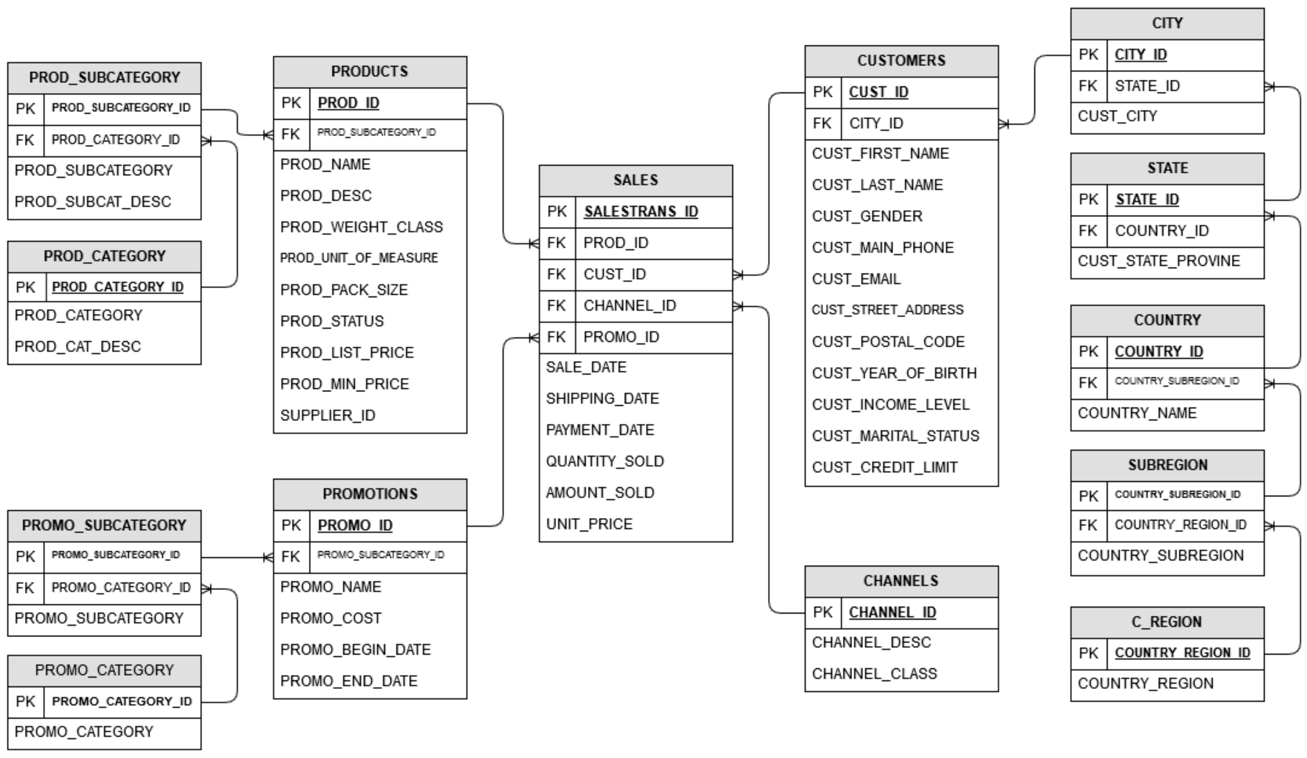


Figure 3. *Entity-Relationship Diagram*

Integrity Constraints

Based on the data inconsistencies found in the Data Understanding section, we made the following changes. Each number corresponds to the same problem identifies in the Data Understanding section.

1. The "Electronics" category was inconsistent, as it had two versions "Electronics" and "ELECTRONICS". We fixed this inconsistency by standardizing both to "Electronics".
2. Two promotions had "ad news" as the promotion category, but "ad news" is supposed to be the subcategory. Therefore, we changed the categories with "ad news" to "newspaper" and moved "ad news" into the corresponding subcategory field.

3. In the LI_CHANNELS table the CHANNEL_ID was not consistent because there was a foreign key reference in other tables with a NUMBER datatype. Our team handled this this by changing the datatype to NUMBER.

4. The CUST_MARITAL_STATUS attribute had many inconsistencies as same information was expressed in different forms. Thus, we standardized the attribute values to the following:

   a. Not sure

   b. Divorced

   c. Single

   d. Widowed

   e. Married

In addition, we constrained CUST_MARITAL_STATUS so that it will only accept the above values, which handles the check constraint as well.

5. In the LI_SALES_12_13 and LI_SALES_14 tables all the primary and foreign key constraints are defined as the VARCHAR datatype. We changed these to the NUMBER datatype, as they as they referenced by their parent tables.

6. We also implemented a constraint for the CUST_GENDER attribute. This field will only accept the following values:

   a. M

   b. F

Based on the useless data found in the Data Understanding section, we made the following changes.

1. In the LI_CHANNELS table the column CHANNEL_TOTAL is unnecessary since all of the rows contain the same value "CHANNEL TOTAL". Thus, CHANNEL_TOTAL was eliminated from the database design.
2. In the LI_CUSTOMER_INTX table the columns CUST_TOTAL and COUNTRY_TOTAL are unnecessary since the same value is listed in each column, "CUST_TOTAL" and "COUNTRY_TOTAL", respectively. Both columns were removed from the database design.
3. In the LI_PRODUCTS table the columns PROD_STATUS and PROD_TOTAL are unnecessary since the same value is listed in each column, "PROD_STATUS" and "PROD_TOTAL", respectively. Both columns were removed.
4. In the LI_PROMOTIONS table the column PROMO_TOTAL is unnecessary because the rows contain the same value, "PROMO TOTAL". This column was removed from the database design.

For each base table we defined the following primary key constraints, as well as referential integrity constraints and inter-columns when necessary.

1. TABLE NAME: group3022_promo_category

    PRIMARY KEY: promo_category_id
2. TABLE NAME: group3022_promo_subcategory

    PRIMARY KEY: promo_subcategory_id

    FOREIGN KEY: prod_category_id which references the PRIMARY KEY in table group3022_promo_category
3. TABLE NAME: group3022_promotions

    PRIMARY KEY: promo_id

    FOREIGN KEY: promo_subcategory_id which references the PRIMARY KEY in table group3022_promo_subcategory

4. TABLE NAME: group3022_prod_category

    PRIMARY KEY: prod_category_id

5. TABLE NAME: group3022_prod_subcategory

    PRIMARY KEY: prod_subcategory_id

    FOREIGN KEY: prod_category_id which references the PRIMARY KEY in table group3022_prod_category

6. TABLE NAME: group3022_products

    PRIMARY KEY: prod_id

    FOREIGN KEY: prod_subcategory_id which references the PRIMARY KEY in table group3022_prod_subcategory

7. TABLE NAME: group3022_region

    PRIMARY KEY: country_region_id

8. TABLE NAME: group3022_subregion

    PRIMARY KEY: country_subregion_id

    FOREIGN KEY: country_region_id which references the PRIMARY KEY in table group3022_region

9. TABLE NAME: group3022_country

    PRIMARY KEY: country_id

    FOREIGN KEY: country_region_id which references the PRIMARY KEY in table group3022_subregion

10. TABLE NAME: group3022_state

    PRIMARY KEY: state_id

    FOREIGN KEY: country_id which references the PRIMARY KEY in table group3022_country

11. TABLE NAME: group3022_city

PRIMARY KEY: city_id

FOREIGN KEY: state_id which references the PRIMARY KEY in table group3022_state

12. TABLE NAME: group3022_customers

PRIMARY KEY: cust_id

FOREIGN KEY: city_id which references the PRIMARY KEY in table group3022_city

13. TABLE NAME: group3022_channels

PRIMARY KEY: channel_id

14. TABLE NAME: group3022_sales

PRIMARY KEY: salestrans_id

FOREIGN KEY: prod_id, cust_id, channel_id, promo_id which references the PRIMARY KEY in table group3022_products, group3022_customers, group3022_channels and group3022_promo

Table Creation (Query Profile)

Before creating the final tables represented in the entity-relationship diagram we needed to create views to calculate the storage requirements. Below are the views we created for this purpose.

Views

v_group3022_promo_category.

```
CREATE OR REPLACE VIEW group3022.v_group3022_promo_category AS
SELECT
ROW_NUMBER() OVER (ORDER BY promo.promo_category) AS promo_category_id,
promo.promo_category
FROM (
SELECT DISTINCT CASE WHEN promo_category = 'ad news' THEN 'newspaper'
ELSE promo_category END AS promo_category
FROM liy26.li_promotions
) promo;
```

v_group3022_promo_subcategory.

```
CREATE OR REPLACE VIEW group3022.v_group3022_promo_subcategory AS
SELECT ROW_NUMBER() OVER (ORDER BY subcat.promo_category,
subcat.promo_subcategory) AS promo_subcategory_id, cat.promo_category_id,
subcat.promo_subcategory
FROM (
SELECT DISTINCT
CASE WHEN promo_subcategory = 'NO RPOMOTION' THEN 'NO PROMOTION'
WHEN promo_category = 'ad news' AND promo_subcategory = 'newspaper' THEN 'ad news'
ELSE promo_subcategory END AS promo_subcategory,
CASE WHEN promo_category = 'ad news' THEN 'newspaper' ELSE promo_category end AS
promo_category
FROM liy26.li_promotions promo
) subcat
INNER JOIN group3022.v_group3022_promo_category cat ON subcat.promo_category =
cat.promo_category;
```

v_group3022_promotions.

```
CREATE OR REPLACE VIEW group3022.v_group3022_promotions AS
SELECT DISTINCT promo_fixed.promo_id, subcat.promo_subcategory_id,
promo_fixed.promo_name, promo_fixed.promo_cost, promo_fixed.promo_begin_date,
promo_fixed.promo_end_date
FROM (
SELECT DISTINCT subcat.promo_subcategory_id, cat.promo_category,
subcat.promo_subcategory
FROM group3022.v_group3022_promo_subcategory subcat
INNER JOIN group3022.v_group3022_promo_category cat ON subcat.promo_category_id =
cat.promo_category_id
) subcat
INNER JOIN (
SELECT DISTINCT promo_id,promo_name,promo_cost,
CASE WHEN promo_subcategory = 'NO RPOMOTION' THEN 'NO PROMOTION'
WHEN promo_category = 'ad news' AND promo_subcategory = 'newspaper' THEN 'ad news'
ELSE promo_subcategory END AS promo_subcategory,
CASE WHEN promo_category = 'ad news' THEN 'newspaper'
ELSE promo_category end AS promo_category, promo_begin_date,promo_end_date
FROM liy26.li_promotions promo
) promo_fixed ON promo_fixed.promo_category = subcat.promo_category
AND promo_fixed.promo_subcategory = subcat.promo_subcategory;
```

v_group3022_prod_category.

```
CREATE OR REPLACE VIEW group3022.v_group3022_prod_category AS
SELECT ROW_NUMBER() OVER (ORDER BY prod_category,prod_cat_desc) AS
prod_category_id, prod_category,prod_cat_desc
FROM
(SELECT DISTINCT CASE WHEN prod_category = 'ELECTRONICS' THEN 'Photo' ELSE
prod_category END AS prod_category,prod_cat_desc
FROM liy26.li_products);
```

v_group3022_prod_subcategory.

```
CREATE OR REPLACE VIEW group3022.v_group3022_prod_subcategory AS
SELECT ROW_NUMBER() OVER (ORDER BY cat.prod_category_id, prod.prod_subcategory,
prod.prod_subcat_desc) AS prod_subcategory_id, cat.prod_category_id, prod.prod_subcategory,
prod.prod_subcat_desc
FROM (
SELECT DISTINCT CASE WHEN prod_category = 'ELECTRONICS' THEN 'Photo' ELSE
prod_category END AS prod_category, prod_subcategory,prod_subcat_desc
FROM liy26.li_products) prod
INNER JOIN group3022.v_group3022_prod_category cat ON prod.prod_category =
cat.prod_category;
```

v_group3022_products.

```
CREATE OR REPLACE VIEW group3022.v_group3022_products AS
SELECT prod.prod_id, subcat_cat.prod_subcategory_id, prod.prod_name, prod.prod_desc,
prod.prod_weight_class, prod.prod_unit_of_measure, prod.prod_pack_size, prod.supplier_id,
prod.prod_list_price,prod.prod_min_price
FROM (SELECT DISTINCT prod_id, CASE WHEN prod_category = 'ELECTRONICS' THEN
'Photo' ELSE prod_category END AS prod_category,prod_cat_desc, prod_subcategory,
prod_subcat_desc, prod_name, prod_desc, prod_weight_class, prod_unit_of_measure,
prod_pack_size, supplier_id, prod_list_price,prod_min_price
FROM liy26.li_products) prod
INNER JOIN (
SELECT subcat.prod_subcategory_id, cat.prod_category, cat.prod_cat_desc,
subcat.prod_subcategory, subcat.prod_subcat_desc
FROM group3022.v_group3022_prod_subcategory subcat
INNER JOIN group3022.v_group3022_prod_category cat ON subcat.prod_category_id =
cat.prod_category_id
) subcat_cat
ON prod.prod_category = subcat_cat.prod_category
AND prod.prod_cat_desc = subcat_cat.prod_cat_desc
AND prod.prod_subcategory = subcat_cat.prod_subcategory
AND prod.prod_subcat_desc = subcat_cat.prod_subcat_desc;
```

v_group3022_region.

```
CREATE OR REPLACE VIEW group3022.v_group3022_region AS
SELECT ROW_NUMBER() OVER (ORDER BY country_region) AS
country_region_id,country_region
FROM (
SELECT distinct country_region
FROM liy26.li_customers_intx);
```

v_group3022_subregion.

```
CREATE OR REPLACE VIEW group3022.v_group3022_subregion AS
```

```
SELECT ROW_NUMBER() OVER (
ORDER BY b.country_region_id, a.country_subregion) AS country_subregion_id,
b.country_region_id, a.country_subregion
FROM (
SELECT DISTINCT country_subregion , country_region
FROM liy26.li_customers_intx) a
INNER JOIN group3022.v_group3022_region b ON a.country_region = b.country_region;
```

v_group3022_country.

```
CREATE OR REPLACE VIEW group3022.v_group3022_country AS
SELECT ROW_NUMBER() OVER (ORDER BY b.country_subregion_id, a.country_name) AS
country_id, b.country_subregion_id, a.country_name
FROM (
SELECT DISTINCT country_name, country_subregion
FROM liy26.li_customers_intx) a
INNER JOIN group3022.v_group3022_subregion b ON a.country_subregion =
b.country_subregion;
```

v_group3022_state.

```
CREATE OR REPLACE VIEW group3022.v_group3022_state AS
SELECT ROW_NUMBER() OVER (ORDER BY b.country_id, a.cust_state_province) AS
state_id , b.country_id, a.cust_state_province
FROM (
SELECT DISTINCT cust_state_province, country_name
FROM liy26.li_customers_intx) a
INNER JOIN group3022.v_group3022_country b ON a.country_name = b.country_name;
```

v_group3022 _city.

```
CREATE OR REPLACE VIEW group3022.v_group3022_city AS
SELECT ROW_NUMBER() OVER (ORDER BY st.state_id, int_cust.cust_city) AS city_id,
st.state_id, int_cust.cust_city
```

```
FROM (
SELECT DISTINCT cust_city, cust_state_province
FROM liy26.li_customers_intx) int_cust
INNER JOIN group3022.v_group3022_state st ON int_cust.cust_state_province =
st.cust_state_province;
```

v_group3022_customers.

```
CREATE OR REPLACE VIEW group3022.v_group3022_customers AS
SELECT cust.cust_id, city.city_id, cust.cust_first_name, cust.cust_last_name, cust.cust_gender,
cust.cust_main_phone_number, cust.cust_email, cust.cust_street_address, cust.cust_postal_code,
cust.cust_year_of_birth, cust.cust_marital_status , cust.cust_income_level, cust.cust_credit_limit
FROM (
select distinct intx.cust_id, intx.cust_first_name, intx.cust_last_name, lower(intx.cust_gender)
cust_gender, intx.cust_main_phone_number, intx.cust_email, intx.cust_street_address,
intx.cust_postal_code, ext.cust_year_of_birth
        ,case
            when ext.cust_marital_status in (
                    'single'
                    ,'NeverM'
                    )
                then 'single'
            when ext.cust_marital_status in (
                    'married'
                    ,'Married'
                    )
                then 'married'
            when ext.cust_marital_status in (
                    'widow'
                    ,'Widowed'
                    )
                then 'widowed'
            when ext.cust_marital_status in (
                    '-'
```

```
                                                ,'Mabsent'
                                                ,'Mar-AF'
                                                )
                                        then 'not sure'
                                when ext.cust_marital_status in (
                                                'Separ'
                                                ,'Divorc'
                                                ,'divorced'
                                                )
                                        then 'divorced'
                                else 'not sure'
                                end as cust_marital_status
                        ,ext.cust_income_level
                        ,ext.cust_credit_limit
                        ,intx.cust_city
                from liy26.li_customers_intx intx
                inner join liy26.li_customers_ext ext on intx.cust_id = ext.cust_id
                ) cust
        inner join group3022.v_group3022_city city on cust.cust_city = city.cust_city;
```

v_group3022 _city

```
CREATE OR REPLACE VIEW group3022.v_group3022_channels AS
SELECT channel_id ,channel_desc, lower(channel_class)
FROM liy26.li_channels;
```

v_group3022_sales

```
CREATE OR REPLACE VIEW group3022.v_group3022_sales AS
```

```
SELECT DISTINCT salestrans_id, cast(prod_id as number) as prod_id, cast(cust_id as number) as
cust_id, cast(channel_id as number) as channel_id, cast(promo_id as number) as promo_id, sale_date,
shipping_date,
                ,payment_date
                ,quantity_sold
                ,amount_sold
                ,unit_price
        from liy26.li_sales_12_13
        union
        select distinct salestrans_id
                ,prod_id
                ,cust_id
                ,channel_id
                ,promo_id
                ,sale_date
                ,shipping_date
                ,payment_date
                ,quantity_sold
                ,amount_sold
                ,unit_price
        from liy26.li_sales_14;
```

Storage Calculation for the Above Views

Storage requirements for all tables were calculated based on the above views.

v_group3022_promo_category

```
SELECT
AVG( 3 + 1 + VSIZE(PROMO_CATEGORY_ID) +
1 + VSIZE(PROMO_CATEGORY)
)
FROM GROUP3022.v_group3022_promo_category;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_promo_subcategory

```
SELECT
AVG( 3 + 1 + VSIZE(PROMO_CATEGORY_ID) +
1 + VSIZE(PROMO_SUBCATEGORY)
)
FROM GROUP3022.v_group3022_promo_subcategory;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_promotions

```
SELECT
AVG( 3 + 1 + VSIZE(PROMO _ID) +
1 + VSIZE(PROMO_SUBCATEGORY_ID)
1 + VSIZE(PROMO_NAME)
1 + VSIZE(PROMO_BEGIN_DATE)
1 + VSIZE(PROMO_END_DATE)
)
FROM GROUP3022.v_group3022_promotions;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 32K

v_group3022_prod_category

```
SELECT
AVG( 3 + 1 + VSIZE(PROD_CATEGORY_ID)
1 + VSIZE(PROD_CATEGORY)
1 + VSIZE(PROD_CAT_DESC)
)
FROM GROUP3022.v_group3022_prod_category;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_prod_subcategory

```
SELECT
AVG( 3 + 1 + VSIZE(PROD_SUBCATEGORY_ID)
1 + VSIZE(PROD_SUBCATEGORY)
1 + VSIZE(PROD_SUBCAT_DESC)
)
FROM GROUP3022.v_group3022_prod_subcategory;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_products

```
SELECT
AVG( 3 + 1 + VSIZE(PROD_ID)
1 + VSIZE(PROD_SUBCATEGORY_ID)
1 + VSIZE(PROD_NAME)
1 + VSIZE(PROD_DESC)
1 + VSIZE(PROD_WEIGHT_CLASS)
1 + VSIZE(PROD_UNIT_OF_MEASURE)
1 + VSIZE(PROD_PACK_SIZE)
1 + VSIZE(SUPPLIER_ID)
1 + VSIZE(PROD_LIST_PRICE)
1 + VSIZE(PROD_MIN_PRICE)
)
FROM GROUP3022.v_group3022_products;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_region

```
SELECT
AVG( 3 + 1 + VSIZE(COUNTRY_REGION_ID)
1 + VSIZE(COUNTRY_REGION)
)
FROM GROUP3022.v_group3022_region;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_subregion

```
SELECT
AVG( 3 + 1 + VSIZE(COUNTRY_SUBREGION_ID)
1 + VSIZE(COUNTRY_SUBREGION)
)
FROM GROUP3022.v_group3022_subregion;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:

STORAGE: 8K

v_group3022_country

```
SELECT
AVG( 3 + 1 + VSIZE(COUNTRY _ID)
1 + VSIZE(COUNTRY_SUBREGION_ID)
1 + VSIZE(COUNTRY_NAME)
)
FROM GROUP3022.v_group3022_country;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_state

```
SELECT
AVG( 3 + 1 + VSIZE(STATE _ID)
1 + VSIZE(COUNTRY _ID)
1 + VSIZE(CUST_STATE_PROVINCE)
)
FROM GROUP3022.v_group3022_state;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_city

```
SELECT
AVG( 3 + 1 + VSIZE(CITY _ID)
1 + VSIZE(STATE _ID)
1 + VSIZE(CUST_CITY)
)
FROM GROUP3022.v_group3022_city;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 16K

v_group3022_customers

```
SELECT
AVG( 3 + 1 + VSIZE(CUST _ID)
1 + VSIZE(CITY _ID)
1 + VSIZE(CUST_FIRST_NAME)
1 + VSIZE(CUST_LAST_NAME)
1 + VSIZE(CUST_GENDER)
1 + VSIZE(CUST_MAIN_PHONE_NUMBER)
1 + VSIZE(CUST_EMAIL)
1 + VSIZE(CUST_STREET_ADDRESS)
1 + VSIZE(CUST_POSTAL_CODE)
1 + VSIZE(CUST_YEAR_OF_BIRTH)
1 + VSIZE(CUST_MARITAL_STATUS)
1 + VSIZE(CUST_INCOME_LEVEL)
1 + VSIZE(CUST_CREDIT_LIMIT)

)
FROM GROUP3022.v_group3022_customers;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 7792K

v_group3022_channels

```
SELECT
AVG( 3 + 1 + VSIZE(CHANNEL _ID)
1 + VSIZE(CHANNEL_DESC)
1 + VSIZE(CHANNEL_CLASS)
)
FROM GROUP3022.v_group3022_channels;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 8K

v_group3022_sales

```
SELECT
AVG( 3 + 1 + VSIZE(SALESTRANS_ID)
1 + VSIZE(PROD_ID)
1 + VSIZE(CUST_ID)
1 + VSIZE(CHANNEL_ID)
1 + VSIZE(PROMO_ID)
1 + VSIZE(SALE_ID)
1 + VSIZE(SHIPPING_DATE)
1 + VSIZE(PAYMENT_DATE)
1 + VSIZE(QUANTITY_SOLD)
1 + VSIZE(AMOUNT_SOLD)
1 + VSIZE(UNIT_PRICE)
)
FROM GROUP3022.v_group3022_sales;
```

Based on the VSIZE we ran that query and calculated the storage required for the table which came to:
STORAGE: 40048K

Tables Created

Based on the above storage calculations and the entity-relationship diagram in figure 3, we created the following tables so that each relation would be 3NF. Different parameters were chosen for PCTFREE and PCTUSED, based on the needs of the table. For example, the PROMO_CATEGORY and PROMO_SUBCATEGORY tables require very little PCTFREE. Thus, we set the minimum amount to 5. The PRODUCTS and CUSTOMERS table require a bit more free space, so we set PCTFREE to 10 for these tables. Finally, the SALES table requires the most free space, thus we set PCTFREE to 20. Below are the series of tables that define our schema.

GROUP3022_PROMO_CATEGORY

```
CREATE TABLE GROUP3022.GROUP3022_PROMO_CATEGORY
(
  PROMO_CATEGORY_ID NUMBER ,
PROMO_CATEGORY VARCHAR2(30) NOT NULL,
PRIMARY KEY(PROMO_CATEGORY_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_PROMO_SUBCATEGORY

```
CREATE TABLE GROUP3022.GROUP3022_PROMO_SUBCATEGORY
(
  PROMO_SUBCATEGORY_ID NUMBER ,
PROMO_CATEGORY_ID NUMBER NOT NULL,
PROMO_SUBCATEGORY VARCHAR2(30) NOT NULL,
PRIMARY KEY(PROMO_SUBCATEGORY_ID),
FOREIGN KEY(PROMO_CATEGORY_ID) REFERENCES
GROUP3022.GROUP3022_PROMO_CATEGORY(PROMO_CATEGORY_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_PROMOTIONS

```
CREATE TABLE GROUP3022.GROUP3022_PROMOTIONS
(
  PROMO_ID NUMBER ,
PROMO_SUBCATEGORY_ID NUMBER,
PROMO_NAME VARCHAR2(30) NOT NULL,
PROMO_COST NUMBER(10,2) NOT NULL,
PROMO_BEGIN_DATE DATE NOT NULL,
PROMO_END_DATE DATE NOT NULL,
PRIMARY KEY(PROMO_ID),
FOREIGN KEY(PROMO_SUBCATEGORY_ID) REFERENCES
GROUP3022.GROUP3022_PROMO_SUBCATEGORY(PROMO_SUBCATEGORY_ID),
CHECK (PROMO_END_DATE >= PROMO_BEGIN_DATE)
)
STORAGE
(
  INITIAL 32K NEXT 32K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_PROD_CATEGORY

```
CREATE TABLE GROUP3022.GROUP3022_PROD_CATEGORY
(
  PROD_CATEGORY_ID NUMBER ,
PROD_CATEGORY VARCHAR2(50) NOT NULL,
PROD_CAT_DESC VARCHAR2(2000) NOT NULL,
PRIMARY KEY(PROD_CATEGORY_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_PROD_SUBCATEGORY

```
CREATE TABLE GROUP3022.GROUP3022_PROD_SUBCATEGORY
(
  PROD_SUBCATEGORY_ID NUMBER,
PROD_CATEGORY_ID NUMBER NOT NULL,
PROD_SUBCATEGORY VARCHAR2(50) NOT NULL,
PROD_SUBCAT_DESC VARCHAR2(2000) NOT NULL,
PRIMARY KEY(PROD_SUBCATEGORY_ID),
FOREIGN KEY(PROD_CATEGORY_ID) REFERENCES
GROUP3022.GROUP3022_PROD_CATEGORY(PROD_CATEGORY_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_PRODUCTS

```
CREATE TABLE GROUP3022.GROUP3022_PRODUCTS
(
  PROD_ID NUMBER ,
PROD_SUBCATEGORY_ID NUMBER NOT NULL,
PROD_NAME VARCHAR2(50) NOT NULL,
PROD_DESC VARCHAR2(400) NOT NULL,
PROD_WEIGHT_CLASS NUMBER(2,0),
PROD_UNIT_OF_MEASURE VARCHAR2(20),
PROD_PACK_SIZE VARCHAR2(30),
SUPPLIER_ID NUMBER(6,0),
PROD_LIST_PRICE NUMBER(8,2) NOT NULL,
PROD_MIN_PRICE NUMBER(8,2) NOT NULL,
PRIMARY KEY(PROD_ID),
FOREIGN KEY(PROD_SUBCATEGORY_ID) REFERENCES
GROUP3022.GROUP3022_PROD_SUBCATEGORY(PROD_SUBCATEGORY_ID),
CHECK(PROD_LIST_PRICE >= PROD_MIN_PRICE)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 10 PCTUSED 20 INITRANS 1;
```

GROUP3022_REGION

```
CREATE TABLE GROUP3022.GROUP3022_REGION
(
  COUNTRY_REGION_ID NUMBER ,
COUNTRY_REGION VARCHAR2(20),
PRIMARY KEY(COUNTRY_REGION_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_SUBREGION

```
CREATE TABLE GROUP3022.GROUP3022_SUBREGION
(
  COUNTRY_SUBREGION_ID NUMBER ,
COUNTRY_REGION_ID NUMBER,
COUNTRY_SUBREGION VARCHAR2(20),
PRIMARY KEY(COUNTRY_SUBREGION_ID),
FOREIGN KEY(COUNTRY_REGION_ID) REFERENCES
GROUP3022.GROUP3022_REGION(COUNTRY_REGION_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_COUNTRY

```
CREATE TABLE GROUP3022.GROUP3022_COUNTRY
(
  COUNTRY_ID NUMBER,
COUNTRY_SUBREGION_ID NUMBER,
COUNTRY_NAME VARCHAR2(50),
PRIMARY KEY(COUNTRY_ID),
FOREIGN KEY(COUNTRY_SUBREGION_ID) REFERENCES
GROUP3022.GROUP3022_SUBREGION(COUNTRY_SUBREGION_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_STATE

```
CREATE TABLE GROUP3022.GROUP3022_STATE
(
  STATE_ID NUMBER ,
COUNTRY_ID NUMBER,
CUST_STATE_PROVINCE VARCHAR2(40),
PRIMARY KEY(STATE_ID),
FOREIGN KEY(COUNTRY_ID) REFERENCES
GROUP3022.GROUP3022_COUNTRY(COUNTRY_ID)
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_CITY

```
CREATE TABLE GROUP3022.GROUP3022_CITY
(
  CITY_ID NUMBER ,
STATE_ID NUMBER,
CUST_CITY VARCHAR2(30) NOT NULL,
PRIMARY KEY(CITY_ID),
FOREIGN KEY(STATE_ID) REFERENCES GROUP3022.GROUP3022_STATE(STATE_ID)
)
STORAGE
(
  INITIAL 16K NEXT 16K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_CUSTOMERS

```
CREATE TABLE GROUP3022.GROUP3022_CUSTOMERS
(
  CUST_ID NUMBER,
CITY_ID NUMBER NOT NULL,
CUST_FIRST_NAME VARCHAR2(20) NOT NULL,
CUST_LAST_NAME VARCHAR2(40) NOT NULL,
CUST_GENDER CHAR(1),
CUST_MAIN_PHONE_NUMBER VARCHAR2(25),
CUST_EMAIL VARCHAR2(30),
CUST_STREET_ADDRESS VARCHAR2(40) NOT NULL,
CUST_POSTAL_CODE  VARCHAR2(10) NOT NULL,
CUST_YEAR_OF_BIRTH NUMBER(4,0),
CUST_MARITAL_STATUS VARCHAR2(20),
CUST_INCOME_LEVEL VARCHAR2(30),
CUST_CREDIT_LIMIT NUMBER,
PRIMARY KEY(CUST_ID),
FOREIGN KEY(CITY_ID) REFERENCES GROUP3022.GROUP3022_CITY(CITY_ID),
CHECK(CUST_GENDER IN ('m' , 'f' , 'o')),
CHECK(CUST_MARITAL_STATUS IN ('not sure', 'divorced', 'single', 'widowed', 'married'))
)
```

```
STORAGE
(
  INITIAL 7792K NEXT 7792K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 10 PCTUSED 20 INITRANS 1;
```

GROUP3022_CHANNELS

```
CREATE TABLE GROUP3022.GROUP3022_CHANNELS
(
CHANNEL_ID NUMBER NOT NULL,
CHANNEL_DESC VARCHAR2(20) NOT NULL,
CHANNEL_CLASS VARCHAR2(20),
PRIMARY KEY(CHANNEL_ID),
CHECK(CHANNEL_CLASS IN ('direct','indirect','others'))
)
STORAGE
(
  INITIAL 8K NEXT 8K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 5 PCTUSED 90 INITRANS 1;
```

GROUP3022_SALES

```
CREATE TABLE GROUP3022.GROUP3022_SALES
(
  SALESTRANS_ID NUMBER ,
PROD_ID NUMBER,
CUST_ID NUMBER,
CHANNEL_ID NUMBER,
PROMO_ID NUMBER,
SALE_DATE DATE,
SHIPPING_DATE DATE,
PAYMENT_DATE DATE ,
QUANTITY_SOLD NUMBER,
AMOUNT_SOLD NUMBER,
UNIT_PRICE NUMBER,
PRIMARY KEY(SALESTRANS_ID),
FOREIGN KEY(PROD_ID) REFERENCES
GROUP3022.GROUP3022_PRODUCTS(PROD_ID),
FOREIGN KEY(CUST_ID) REFERENCES
GROUP3022.GROUP3022_CUSTOMERS(CUST_ID),
FOREIGN KEY(CHANNEL_ID) REFERENCES
GROUP3022.GROUP3022_CHANNELS(CHANNEL_ID),
FOREIGN KEY(PROMO_ID) REFERENCES
GROUP3022.GROUP3022_PROMOTIONS(PROMO_ID),
CHECK(SHIPPING_DATE >= SALE_DATE),
CHECK(PAYMENT_DATE >= SALE_DATE)
)
STORAGE
(
  INITIAL 40048K NEXT 40048K
  MINEXTENTS 1 PCTINCREASE 5
)
PCTFREE 20 PCTUSED 10 INITRANS 4;
```

Inserting Data into Tables

While inserting the data into the tables created above we have handled all the **check constraints** and **integrity constraints** so that we do not break any relations inside the tables.

Insert into the group3022_promo_category table

```
insert into group3022.group3022_promo_category select * from
group3022.v_group3022_promo_category;
```

Insert into the group3022_promo_subcategory table

```
insert into group3022.group3022_promo_subcategory select * from
group3022.v_group3022_promo_subcategory;
```

Insert into the group3022_promotions table

```
insert into group3022.group3022_promotions select * from group3022.v_group3022_promotions;
```

Insert into the group3022_prod_category table

```
insert into group3022.group3022_prod_category select * from
group3022.v_group3022_prod_category;
```

Insert into the group3022_prod_subcategory table

```
insert into group3022.group3022_prod_subcategory select * from
group3022.v_group3022_prod_subcategory;
```

Insert into the group3022_products table

```
insert into group3022.group3022_products select * from group3022.v_group3022_products;
```

Insert into the group3022_region table

```
insert into group3022.group3022_region select * from group3022.v_group3022_region;
```

Insert into the group3022_subregion table

```
insert into group3022.group3022_subregion select * from group3022.v_group3022_subregion;
```

Insert into the group3022_country table

```
insert into group3022.group3022_country select * from group3022.v_group3022_country;
```

Insert into the group3022_state table

```
insert into group3022.group3022_state select * from group3022.v_group3022_state;
```

Insert into the group3022_city table

```
insert into group3022.group3022_city select * from group3022.v_group3022_city;
```

Insert into the group3022_customers table

```
insert into group3022.group3022_customers select * from group3022.v_group3022_customers;
```

Insert into the group3022_channels table

```
insert into group3022.group3022_channels select * from group3022.v_group3022_channels;
```

Insert into the group3022_sales table

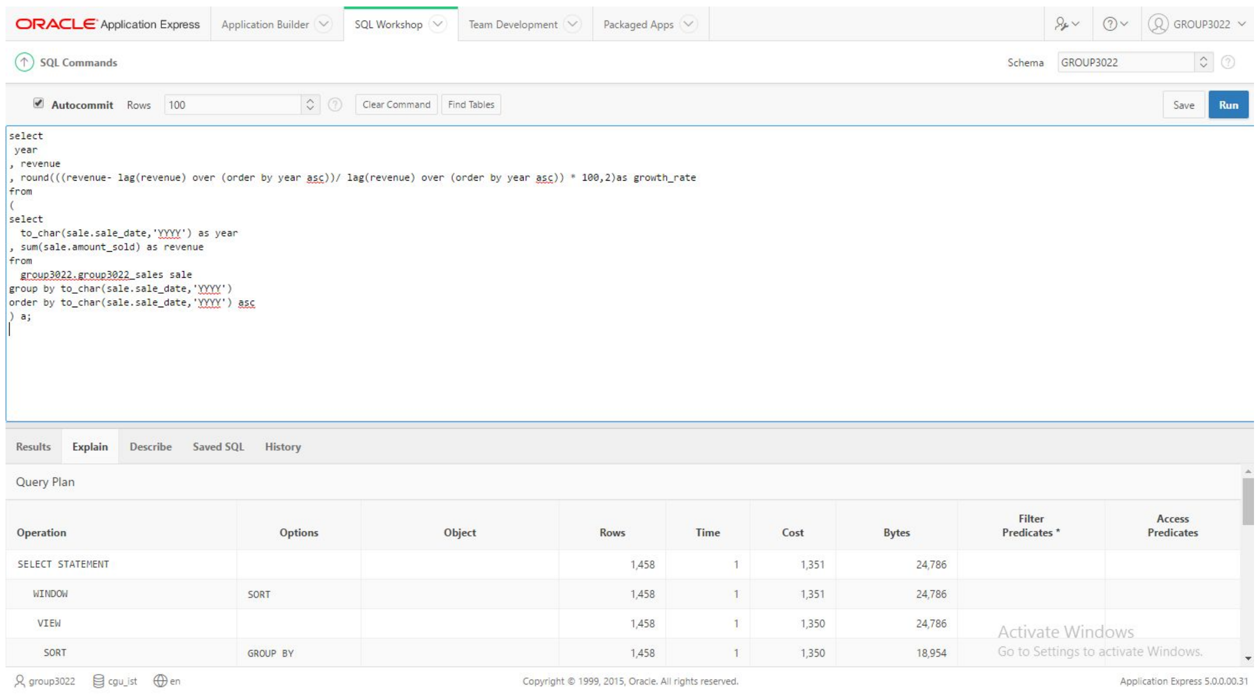
```
insert into group3022.group3022_sales select * from group3022.v_group3022_sales;
```

Access Structures

Accounting Business Views

account_bvq1

Accounting business view execution plan before creating index:


ORACLE Application Express
Application Builder
SQL Workshop
Team Development
Packaged Apps
GROUP3022
SQL Commands
Schema GROUP3022
Autocommit Rows 100
Clear Command
Find Tables
Save
Run
Results Explain Describe Saved SQL History
Activate Windows
Go to Settings to activate Windows.
group3022 cgu_ist en
Copyright © 1999, 2015, Oracle. All rights reserved.
Application Express 5.0.0.00.31

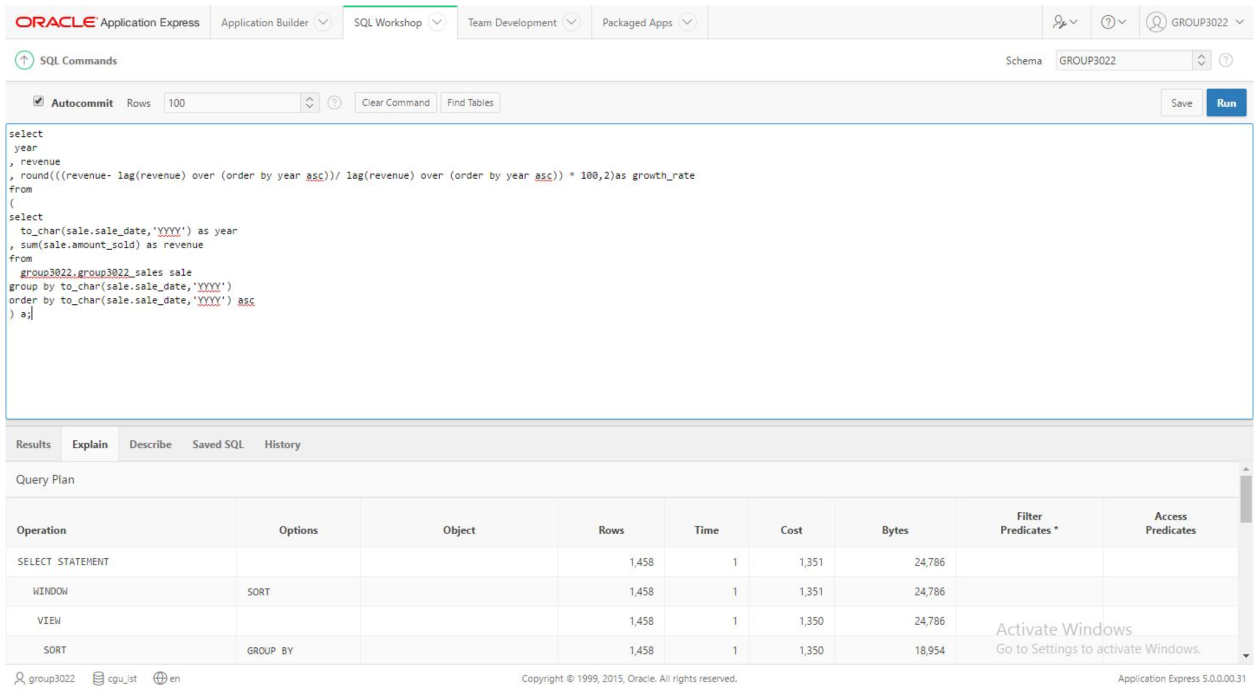
```
select
 year
, revenue
, round(((revenue- lag(revenue) over (order by year asc))/ lag(revenue) over (order by year asc)) * 100,2)as growth_rate
from
(
select
  to_char(sale.sale_date,'YYYY') as year
, sum(sale.amount_sold) as revenue
from
  group3022.group3022_sales sale
group by to_char(sale.sale_date,'YYYY')
order by to_char(sale.sale_date,'YYYY') asc
) a;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1,458	1	1,351	24,786		
WINDOW	SORT		1,458	1	1,351	24,786		
VIEW			1,458	1	1,350	24,786		
SORT	GROUP BY		1,458	1	1,350	18,954		

COST: 1351

Accounting business view execution plan after creating index:


ORACLE Application Express
Application Builder
SQL Workshop
Team Development
Packaged Apps
GROUP3022
SQL Commands
Schema GROUP3022
Autocommit Rows 100
Clear Command
Find Tables
Save
Run
Results Explain Describe Saved SQL History
Activate Windows
Go to Settings to activate Windows.
group3022 cgu_ist en
Copyright © 1999, 2015, Oracle. All rights reserved.
Application Express 5.0.0.00.31

```
select
 year
, revenue
, round(((revenue- lag(revenue) over (order by year asc))/ lag(revenue) over (order by year asc)) * 100,2)as growth_rate
from
(
select
  to_char(sale.sale_date,'YYYY') as year
, sum(sale.amount_sold) as revenue
from
  group3022.group3022_sales sale
group by to_char(sale.sale_date,'YYYY')
order by to_char(sale.sale_date,'YYYY') asc
) a;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1,458	1	1,351	24,786		
WINDOW	SORT		1,458	1	1,351	24,786		
VIEW			1,458	1	1,350	24,786		
SORT	GROUP BY		1,458	1	1,350	18,954		

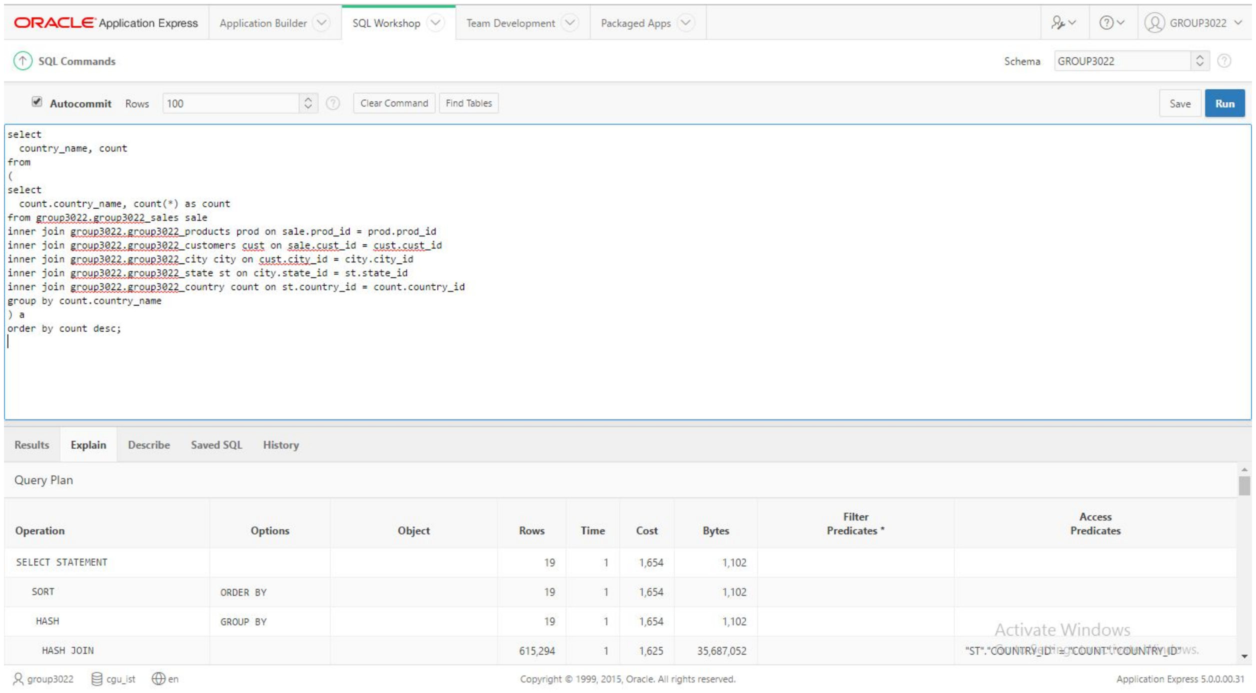
COST:1351

The indexes created did not have any effect on the execution plan of the accounting business view 1 (account_bvq1). Therefore, we will be dropping the indices created.

CEO Business Views

CEO_bvq1

CEO business view plan before creating index:



```
select
  country_name, count
from
(
select
  count.country_name, count(*) as count
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
group by count.country_name
) a
order by count desc;
```

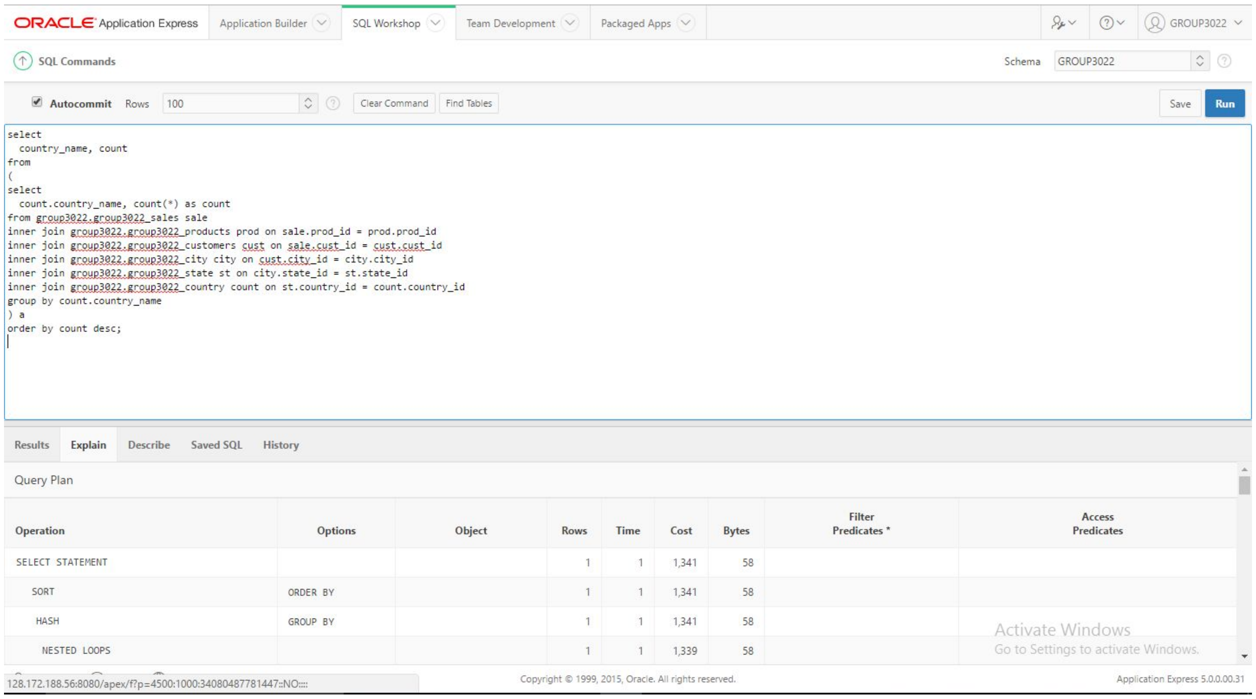
Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			19	1	1,654	1,102		
SORT	ORDER BY		19	1	1,654	1,102		
HASH	GROUP BY		19	1	1,654	1,102		
HASH JOIN			615,294	1	1,625	35,687,052		"ST"."COUNTRY_ID"="COUNT"."COUNTRY_ID"

group3022 cgu_ist en

Application Express 5.0.0.00.31

COST: 1654

CEO business view plan after creating index:



```
select
  country_name, count
from
(
select
  count.country_name, count(*) as count
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
group by count.country_name
) a
order by count desc;
```

Query Plan

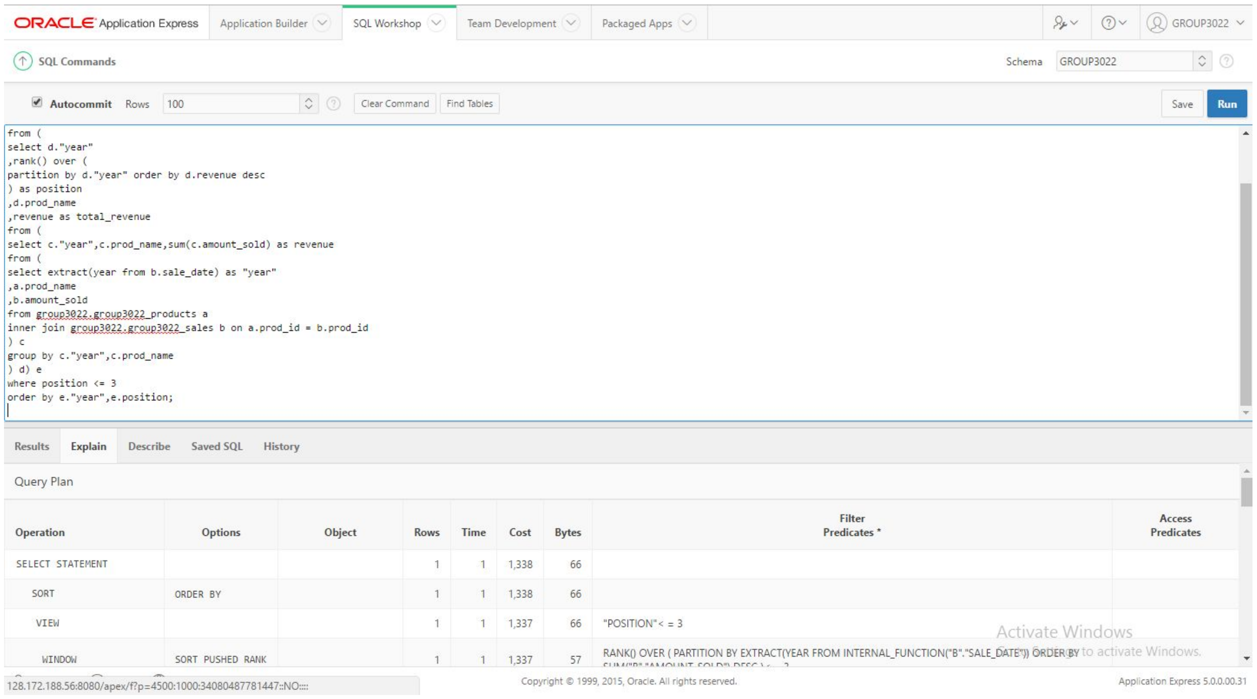
Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,341	58		
SORT	ORDER BY		1	1	1,341	58		
HASH	GROUP BY		1	1	1,341	58		
NESTED LOOPS			1	1	1,339	58		

COST: 1341

The execution plan for this view shows that there is a noticeable change in cost. Hence, we will keep the index for this business view.

CEO_bvq2

CEO business view plan before creating indices:



```
from (
select d."year"
,rank() over (
partition by d."year" order by d.revenue desc
) as position
,d.prod_name
,revenue as total_revenue
from (
select c."year",c.prod_name,sum(c.amount_sold) as revenue
from (
select extract(year from b.sale_date) as "year"
,a.prod_name
,b.amount_sold
from group3022.group3022_products a
inner join group3022.group3022_sales b on a.prod_id = b.prod_id
) c
group by c."year",c.prod_name
) d) e
where position <= 3
order by e."year",e.position;
```

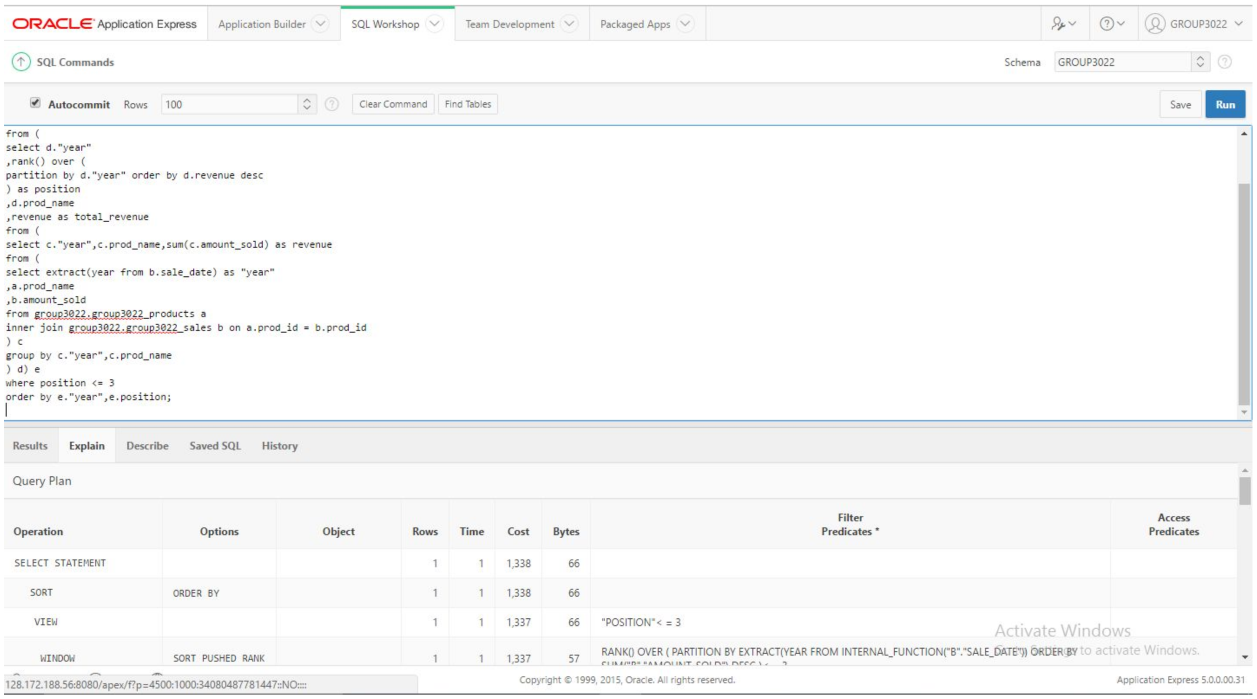
Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,338	66		
SORT	ORDER BY		1	1	1,338	66		
VIEW			1	1	1,337	66	"POSITION"< = 3	
WINDOW	SORT PUSHED RANK		1	1	1,337	57	RANK() OVER (PARTITION BY EXTRACT(YEAR FROM INTERNAL_FUNCTION("B"."SALE_DATE")) ORDER BY	

128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::

Application Express 5.0.0.00.31

COST: 1338

CEO business view plan after creating indices:



```
from (
select d."year"
,rank() over (
partition by d."year" order by d.revenue desc
) as position
,d.prod_name
,revenue as total_revenue
from (
select c."year",c.prod_name,sum(c.amount_sold) as revenue
from (
select extract(year from b.sale_date) as "year"
,a.prod_name
,b.amount_sold
from group3022.group3022_products a
inner join group3022.group3022_sales b on a.prod_id = b.prod_id
) c
group by c."year",c.prod_name
) d) e
where position <= 3
order by e."year",e.position;
```

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,338	66		
SORT	ORDER BY		1	1	1,338	66		
VIEW			1	1	1,337	66	"POSITION"< = 3	
WINDOW	SORT PUSHED RANK		1	1	1,337	57	RANK() OVER (PARTITION BY EXTRACT(YEAR FROM INTERNAL_FUNCTION("B"."SALE_DATE")) ORDER BY	

128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::

Application Express 5.0.0.00.31

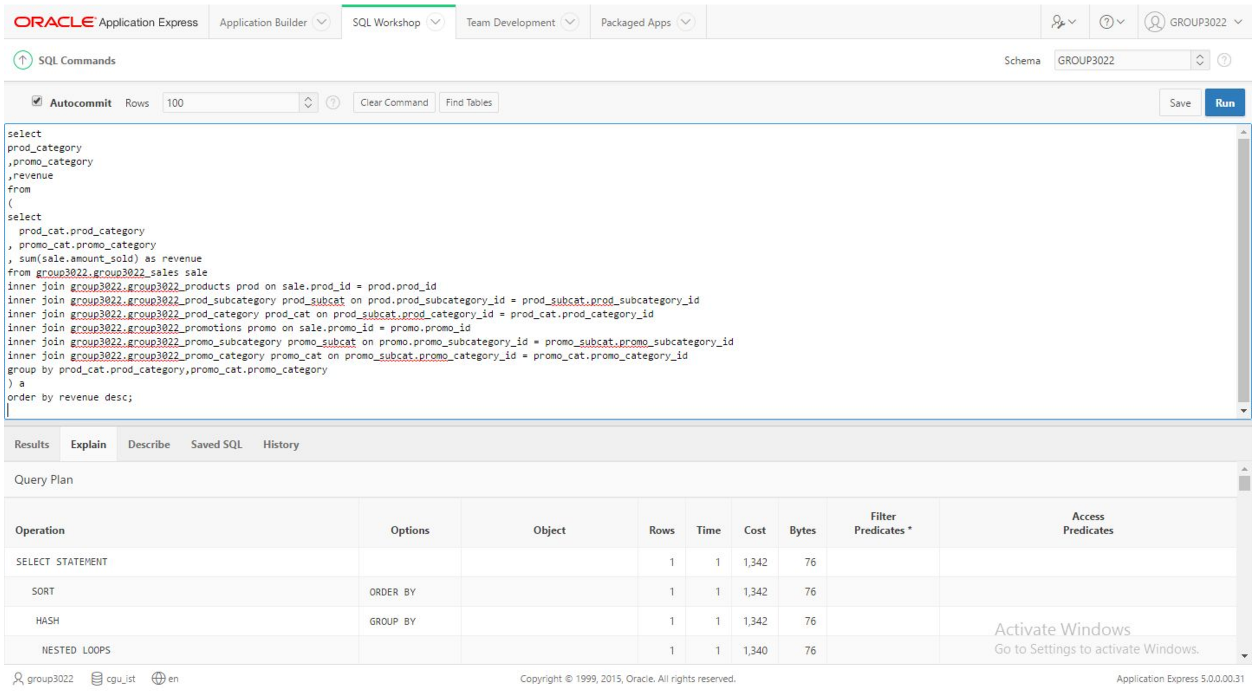
COST: 1338

Since the introduction of indices have no effect in the execution plan we can drop the index as it might be an overhead for the system.

Marketing Business Views

Marketing_bvq1

Marketing business view execution plan before creating index:



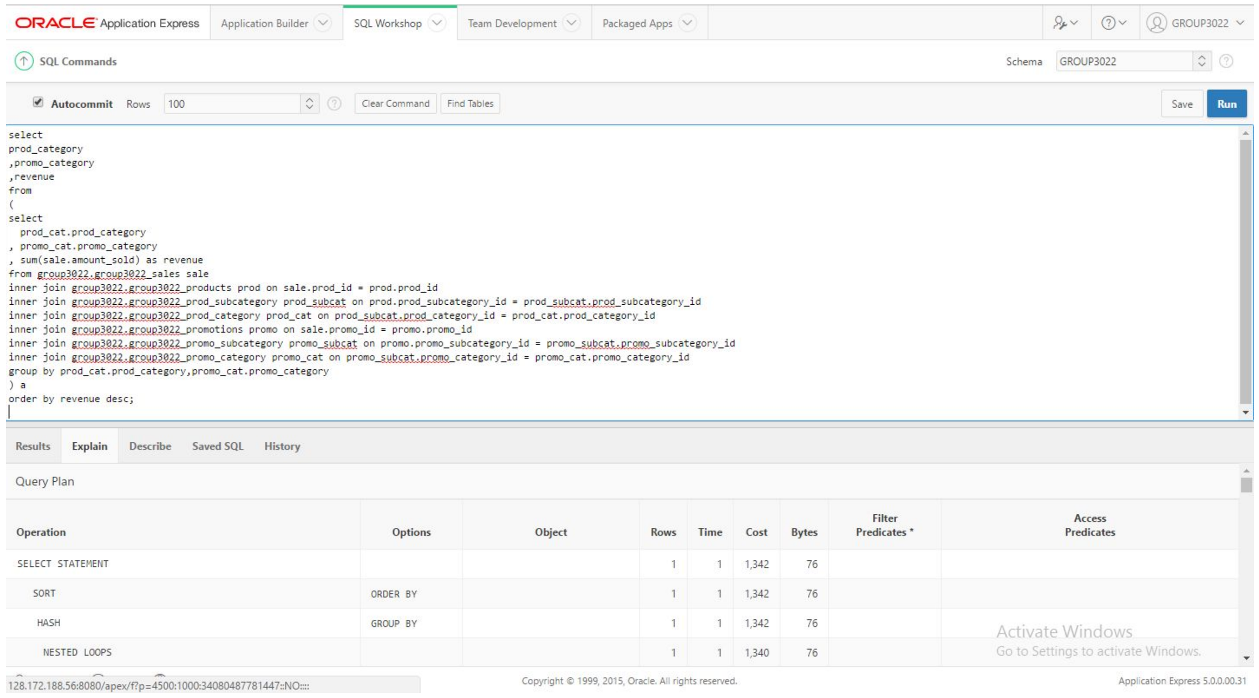
```
select
prod_category
,promo_category
,revenue
from
(
select
  prod_cat.prod_category
, promo_cat.promo_category
, sum(sale.amount_sold) as revenue
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_prod_subcategory prod_subcat on prod.prod_subcategory_id = prod_subcat.prod_subcategory_id
inner join group3022.group3022_prod_category prod_cat on prod_subcat.prod_category_id = prod_cat.prod_category_id
inner join group3022.group3022_promotions promo on sale.promo_id = promo.promo_id
inner join group3022.group3022_promo_subcategory promo_subcat on promo.promo_subcategory_id = promo_subcat.promo_subcategory_id
inner join group3022.group3022_promo_category promo_cat on promo_subcat.promo_category_id = promo_cat.promo_category_id
group by prod_cat.prod_category,promo_cat.promo_category
) a
order by revenue desc;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,342	76		
SORT	ORDER BY		1	1	1,342	76		
HASH	GROUP BY		1	1	1,342	76		
NESTED LOOPS			1	1	1,340	76		

COST: 1342

Marketing business view execution plan after creating index:



```
select
prod_category
,promo_category
,revenue
from
(
select
  prod_cat.prod_category
, promo_cat.promo_category
, sum(sale.amount_sold) as revenue
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_prod_subcategory prod_subcat on prod.prod_subcategory_id = prod_subcat.prod_subcategory_id
inner join group3022.group3022_prod_category prod_cat on prod_subcat.prod_category_id = prod_cat.prod_category_id
inner join group3022.group3022_promotions promo on sale.promo_id = promo.promo_id
inner join group3022.group3022_promo_subcategory promo_subcat on promo.promo_subcategory_id = promo_subcat.promo_subcategory_id
inner join group3022.group3022_promo_category promo_cat on promo_subcat.promo_category_id = promo_cat.promo_category_id
group by prod_cat.prod_category,promo_cat.promo_category
) a
order by revenue desc;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,342	76		
SORT	ORDER BY		1	1	1,342	76		
HASH	GROUP BY		1	1	1,342	76		
NESTED LOOPS			1	1	1,340	76		

128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::

Copyright © 1999, 2015, Oracle. All rights reserved.

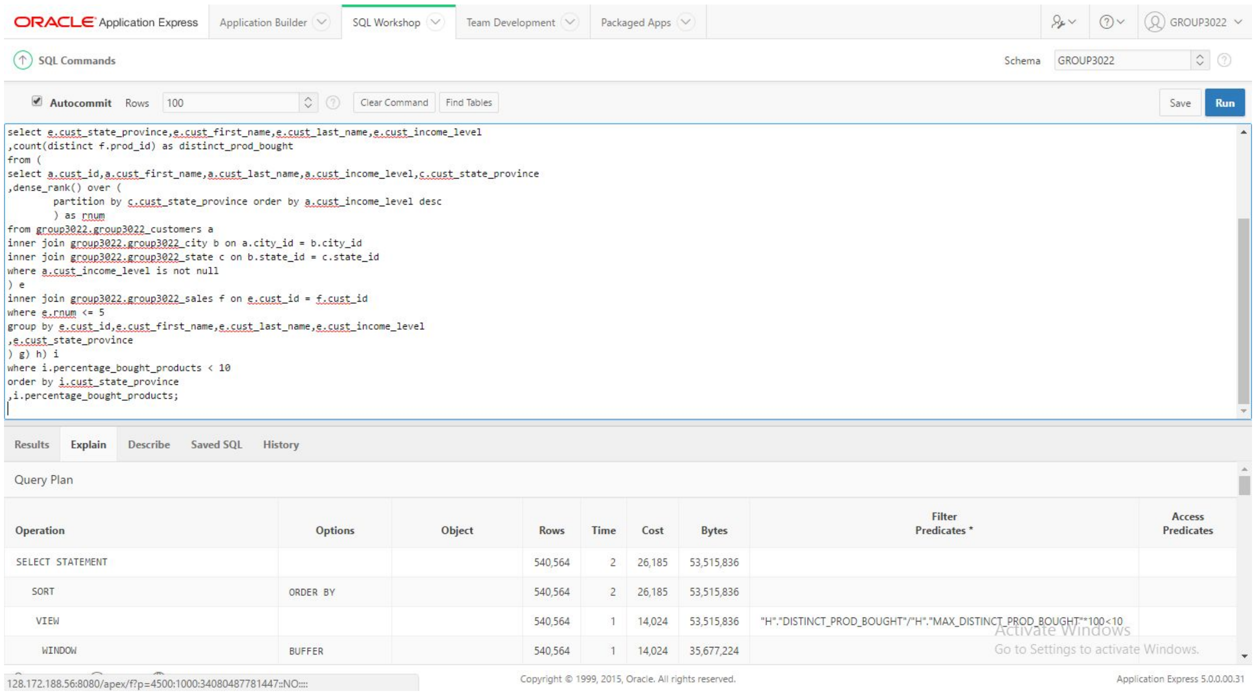
Application Express 5.0.0.00.31

COST: 1342

Since the introduction of indices have no effect in the execution plan we can drop the index as it might be an overhead for the system.

Marketing_bvq3

Marketing business view execution plan before creating index:



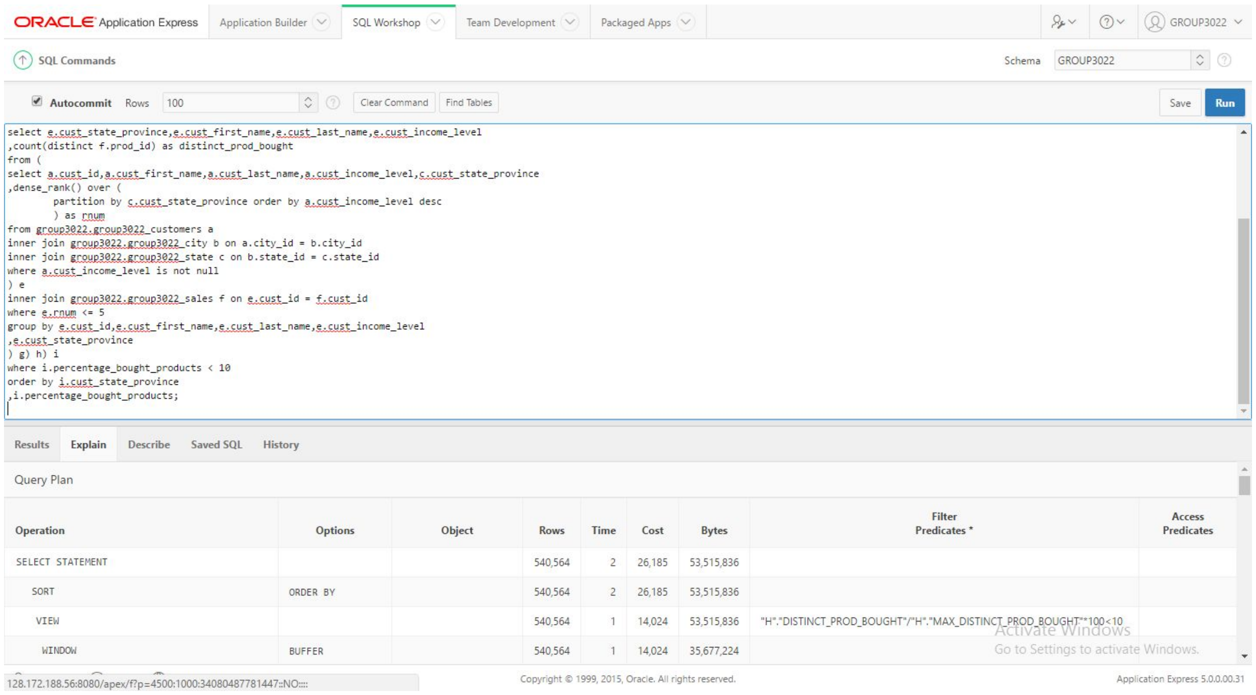
```
select e.cust_state_province,e.cust_first_name,e.cust_last_name,e.cust_income_level
,count(distinct f.prod_id) as distinct_prod_bought
from (
select a.cust_id,a.cust_first_name,a.cust_last_name,a.cust_income_level,c.cust_state_province
,dense_rank() over (
        partition by c.cust_state_province order by a.cust_income_level desc
        ) as rnum
from group3022.group3022_customers a
inner join group3022.group3022_city b on a.city_id = b.city_id
inner join group3022.group3022_state c on b.state_id = c.state_id
where a.cust_income_level is not null
) e
inner join group3022.group3022_sales f on e.cust_id = f.cust_id
where e.rnum <= 5
group by e.cust_id,e.cust_first_name,e.cust_last_name,e.cust_income_level
,e.cust_state_province
) g) h) i
where i.percentage_bought_products < 10
order by i.cust_state_province
,i.percentage_bought_products;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			540,564	2	26,185	53,515,836		
SORT	ORDER BY		540,564	2	26,185	53,515,836		
VIEW			540,564	1	14,024	53,515,836	"H"."DISTINCT_PROD_BOUGHT"/"H"."MAX_DISTINCT_PROD_BOUGHT"*100<10	
WINDOW	BUFFER		540,564	1	14,024	35,677,224		

COST: 26165

Marketing business view execution plan after creating index:



```
select e.cust_state_province,e.cust_first_name,e.cust_last_name,e.cust_income_level
,count(distinct f.prod_id) as distinct_prod_bought
from (
select a.cust_id,a.cust_first_name,a.cust_last_name,a.cust_income_level,c.cust_state_province
,dense_rank() over (
        partition by c.cust_state_province order by a.cust_income_level desc
        ) as rnum
from group3022.group3022_customers a
inner join group3022.group3022_city b on a.city_id = b.city_id
inner join group3022.group3022_state c on b.state_id = c.state_id
where a.cust_income_level is not null
) e
inner join group3022.group3022_sales f on e.cust_id = f.cust_id
where e.rnum <= 5
group by e.cust_id,e.cust_first_name,e.cust_last_name,e.cust_income_level
,e.cust_state_province
) g) h) i
where i.percentage_bought_products < 10
order by i.cust_state_province
,i.percentage_bought_products;
```

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			540,564	2	26,185	53,515,836		
SORT	ORDER BY		540,564	2	26,185	53,515,836		
VIEW			540,564	1	14,024	53,515,836	"H"."DISTINCT_PROD_BOUGHT"/"H"."MAX_DISTINCT_PROD_BOUGHT"*100<10	
WINDOW	BUFFER		540,564	1	14,024	35,677,224		

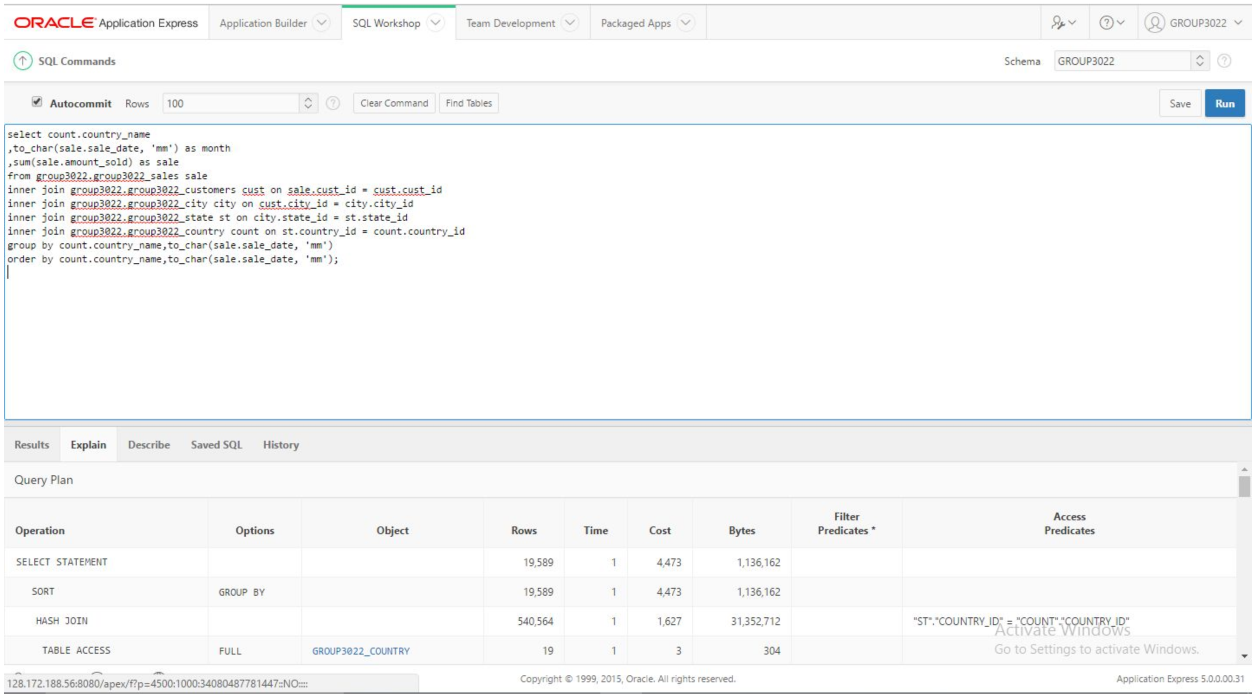
COST: 26165

Since the introduction of indices have no effect in the execution plan we can drop the index as it might be an overhead for the system.

Sales Business Views

Sales_bvq1

Sales business view for execution plan before creating index:



```
select count.country_name
,to_char(sale.sale_date, 'mm') as month
,sum(sale.amount_sold) as sale
from group3022.group3022_sales sale
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
group by count.country_name,to_char(sale.sale_date, 'mm')
order by count.country_name,to_char(sale.sale_date, 'mm');
```

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			19,589	1	4,473	1,136,162		
SORT	GROUP BY		19,589	1	4,473	1,136,162		
HASH JOIN			540,564	1	1,627	31,352,712		"ST"."COUNTRY_ID" = "COUNT"."COUNTRY_ID"
TABLE ACCESS	FULL	GROUP3022_COUNTRY	19	1	3	304		

Activate Windows
Go to Settings to activate Windows.

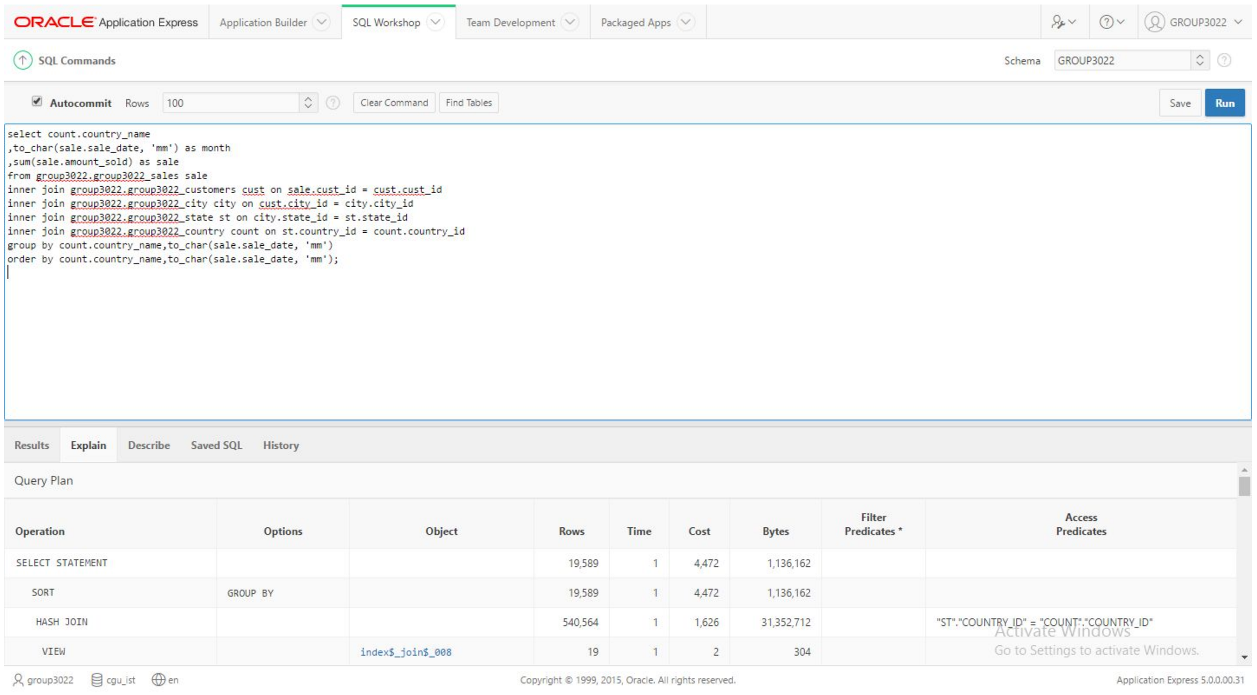
128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::

Copyright © 1999, 2015, Oracle. All rights reserved.

Application Express 5.0.0.00.31

COST: 4473

Sales business view for execution plan after creating index:



```
select count.country_name
,to_char(sale.sale_date, 'mm') as month
,sum(sale.amount_sold) as sale
from group3022.group3022_sales sale
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
group by count.country_name,to_char(sale.sale_date, 'mm')
order by count.country_name,to_char(sale.sale_date, 'mm');
```

Query Plan

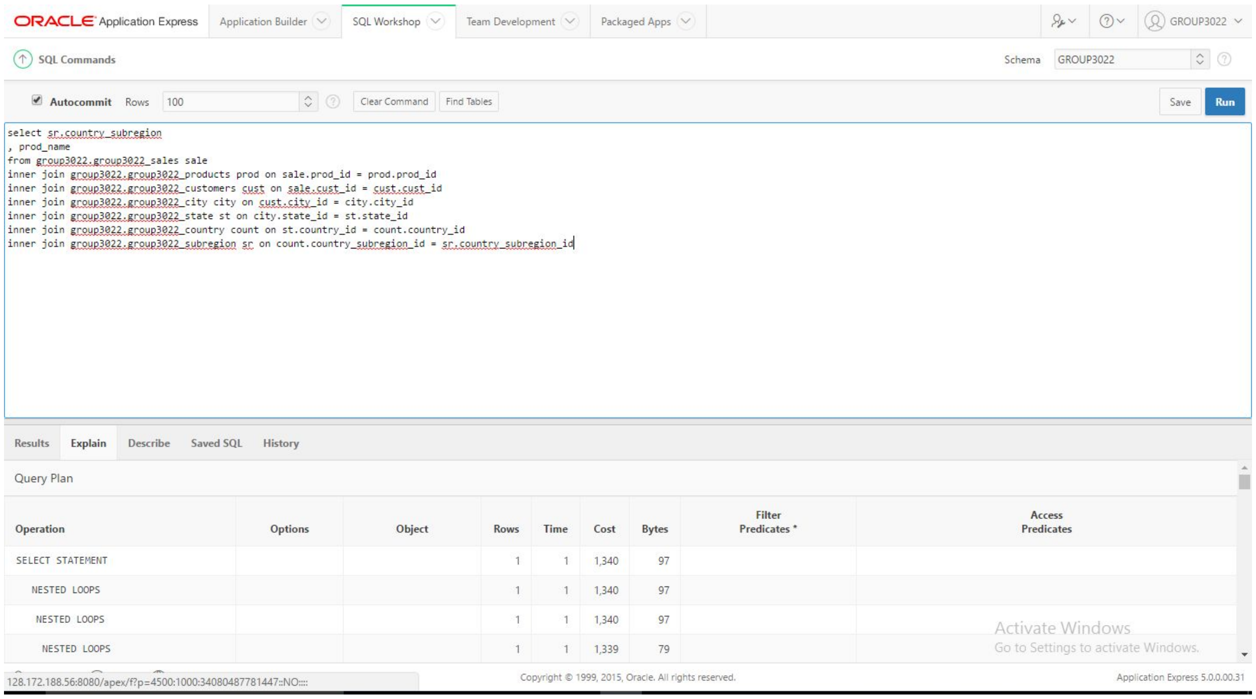
Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			19,589	1	4,472	1,136,162		
SORT	GROUP BY		19,589	1	4,472	1,136,162		
HASH JOIN			540,564	1	1,626	31,352,712		"ST"."COUNTRY_ID" = "COUNT"."COUNTRY_ID"
VIEW		index$_join$_008	19	1	2	304		

COST: 4472

Since there is a slight change in cost after the index creation, we will keep this index.

Sales_bvq2

Sales business view for execution plan before creating index:

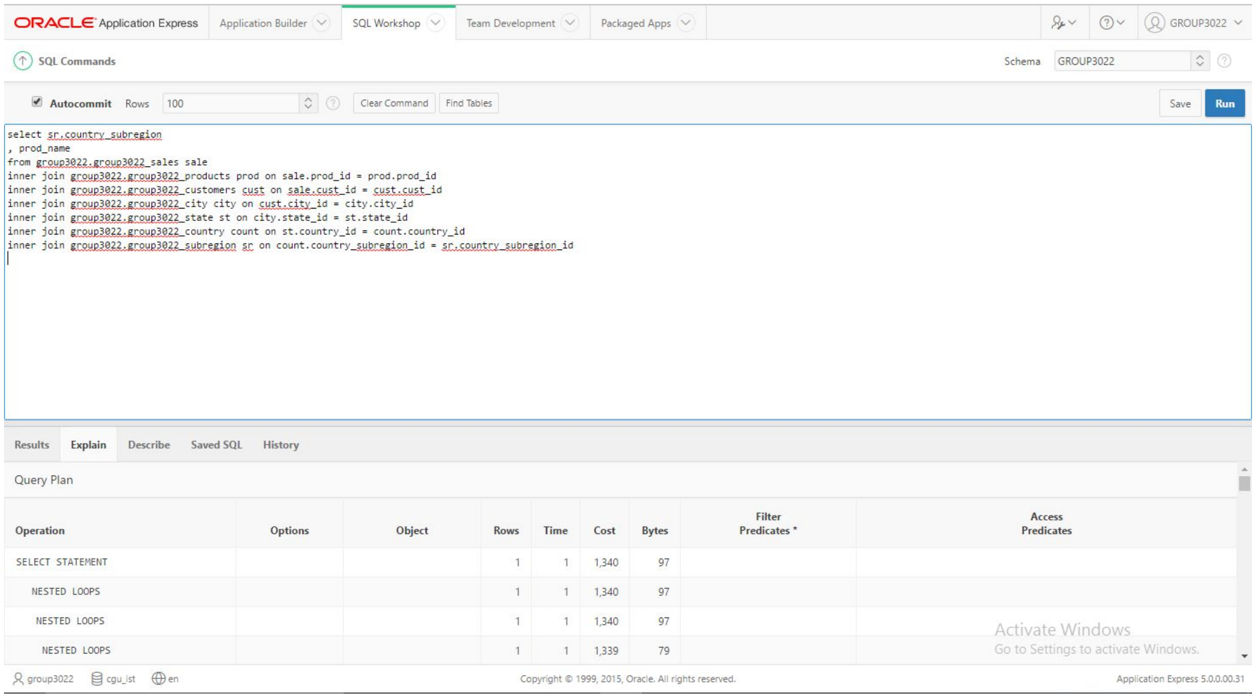


```
select sr.country_subregion
, prod_name
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
inner join group3022.group3022_subregion sr on count.country_subregion_id = sr.country_subregion_id
```

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,340	97		
NESTED LOOPS			1	1	1,340	97		
NESTED LOOPS			1	1	1,340	97		
NESTED LOOPS			1	1	1,339	79		

COST: 1340

Sales business view for execution plan after creating index:



```
select sr.country_subregion
, prod_name
from group3022.group3022_sales sale
inner join group3022.group3022_products prod on sale.prod_id = prod.prod_id
inner join group3022.group3022_customers cust on sale.cust_id = cust.cust_id
inner join group3022.group3022_city city on cust.city_id = city.city_id
inner join group3022.group3022_state st on city.state_id = st.state_id
inner join group3022.group3022_country count on st.country_id = count.country_id
inner join group3022.group3022_subregion sr on count.country_subregion_id = sr.country_subregion_id
```

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	1,340	97		
NESTED LOOPS			1	1	1,340	97		
NESTED LOOPS			1	1	1,340	97		
NESTED LOOPS			1	1	1,339	79		

COST: 1340

Since the introduction of these indices has no effect on the execution plan we can drop this index.

Materialized View

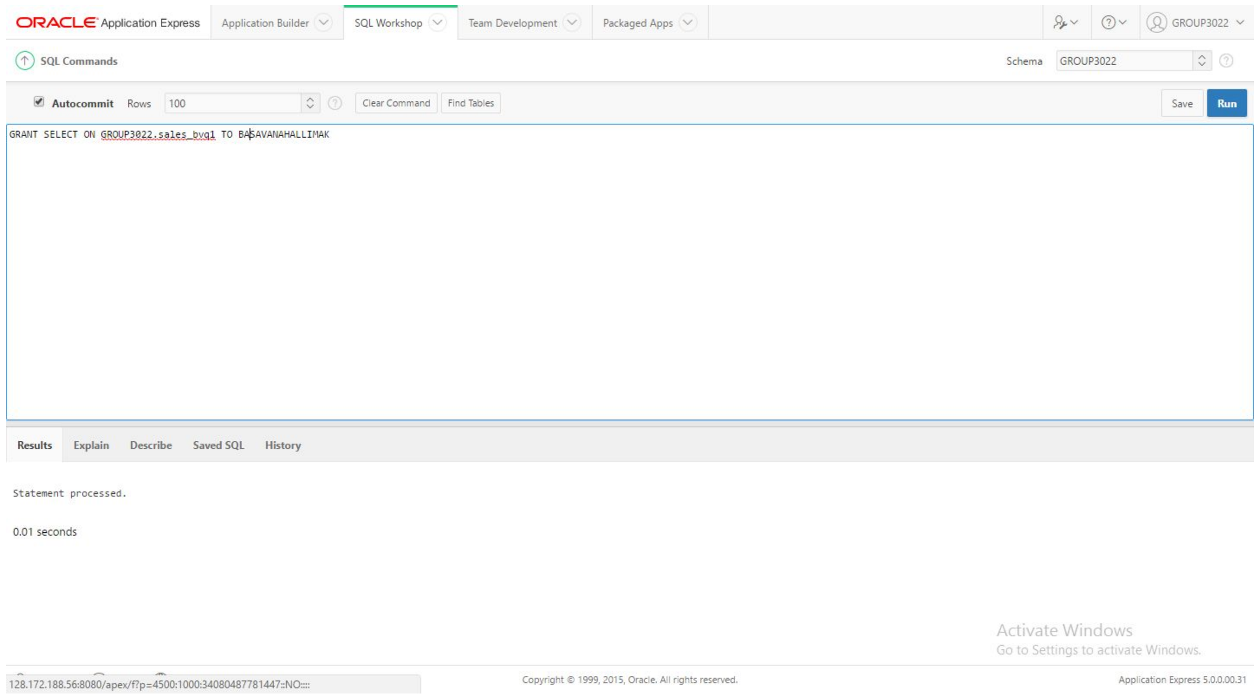
We created the materialized view for the CEO wherein he can get an overview of the products based on country. Since, there won't be much change in the data immediately and he would be interested in viewing it often.

Security Implementation

The following shows the relevant permissions granted to each user.

Sales Director

GRANT SELECT ON GROUP3022.sales_bvq1 TO BASAVANAHALLIMAK

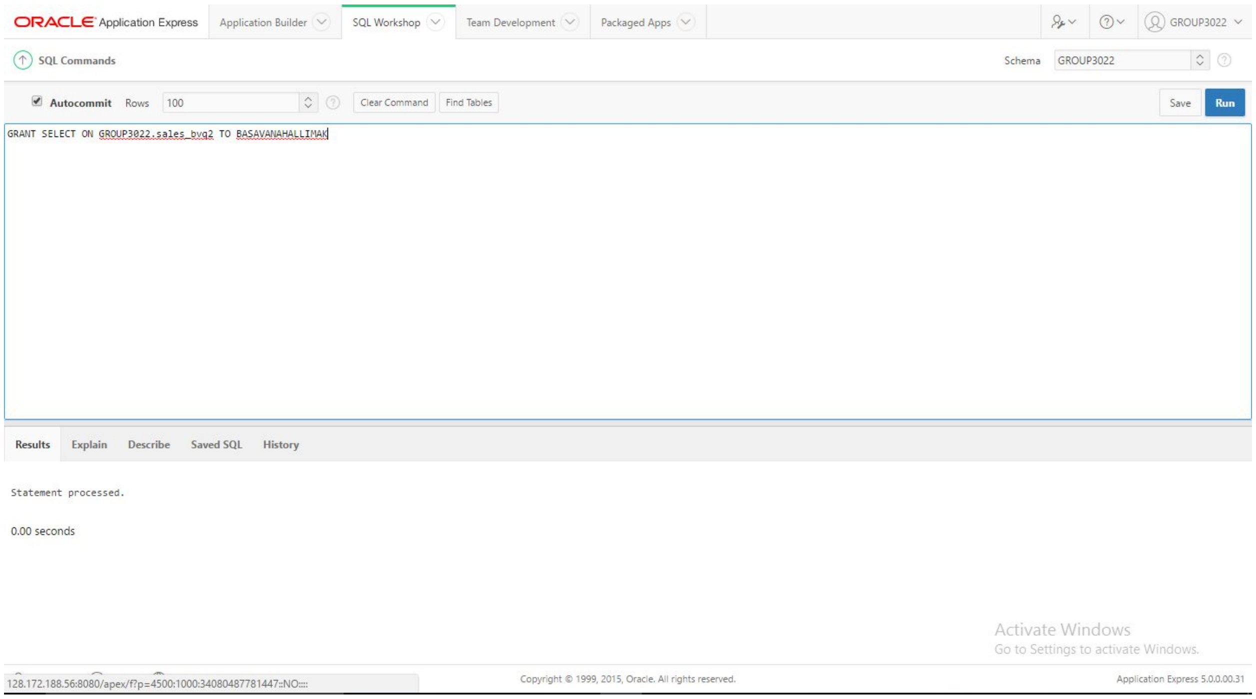


```
GRANT SELECT ON GROUP3022.sales_bvq1 TO BASAVANAHALLIMAK
```

```
Statement processed.

0.01 seconds
```

128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::

GRANT SELECT ON GROUP3022.sales_bvq2 TO BASAVANAHALLIMAK


ORACLE Application Express
Application Builder
SQL Workshop
Team Development
Packaged Apps
GROUP3022
SQL Commands
Schema GROUP3022
Autocommit
Rows 100
Clear Command
Find Tables
Save
Run
Results
Explain
Describe
Saved SQL
History

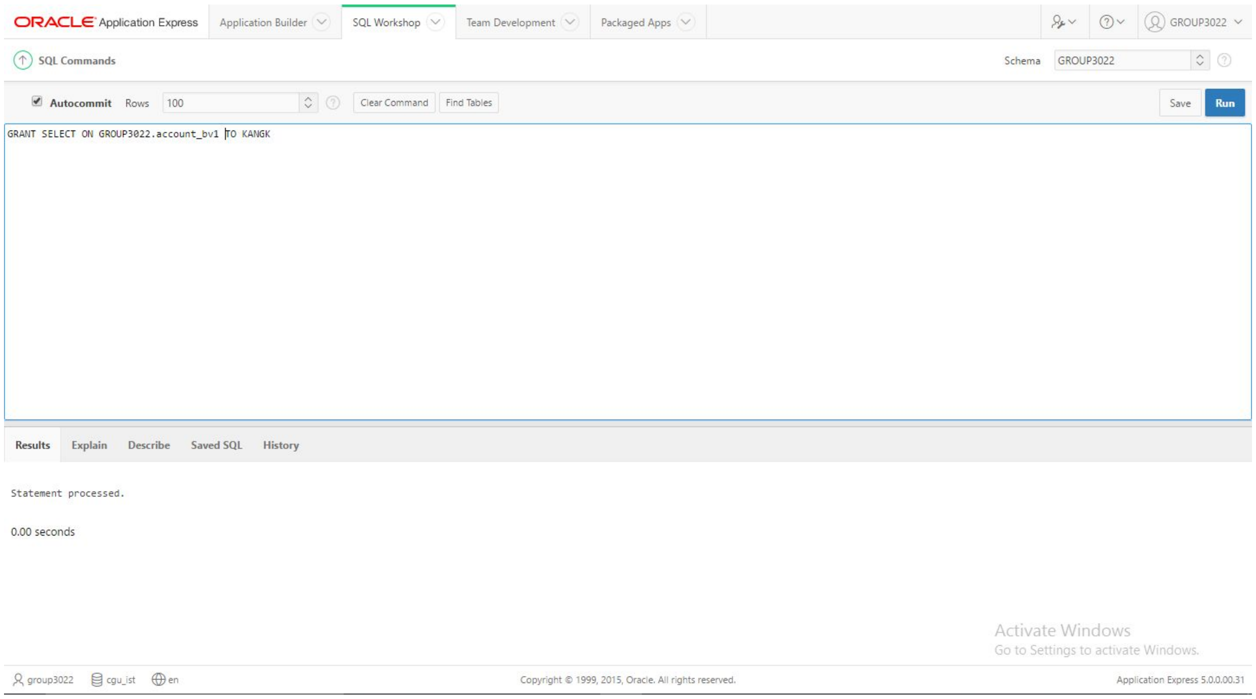
```
GRANT SELECT ON GROUP3022.sales_bvg2 TO BASAVANAHALLIMAK
```

```
Statement processed.

0.00 seconds
```

Activate Windows
Go to Settings to activate Windows.
128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::
Copyright © 1999, 2015, Oracle. All rights reserved.
Application Express 5.0.0.00.31

Accounting Director

GRANT SELECT ON GROUP3022.account_bv1 TO KANGK



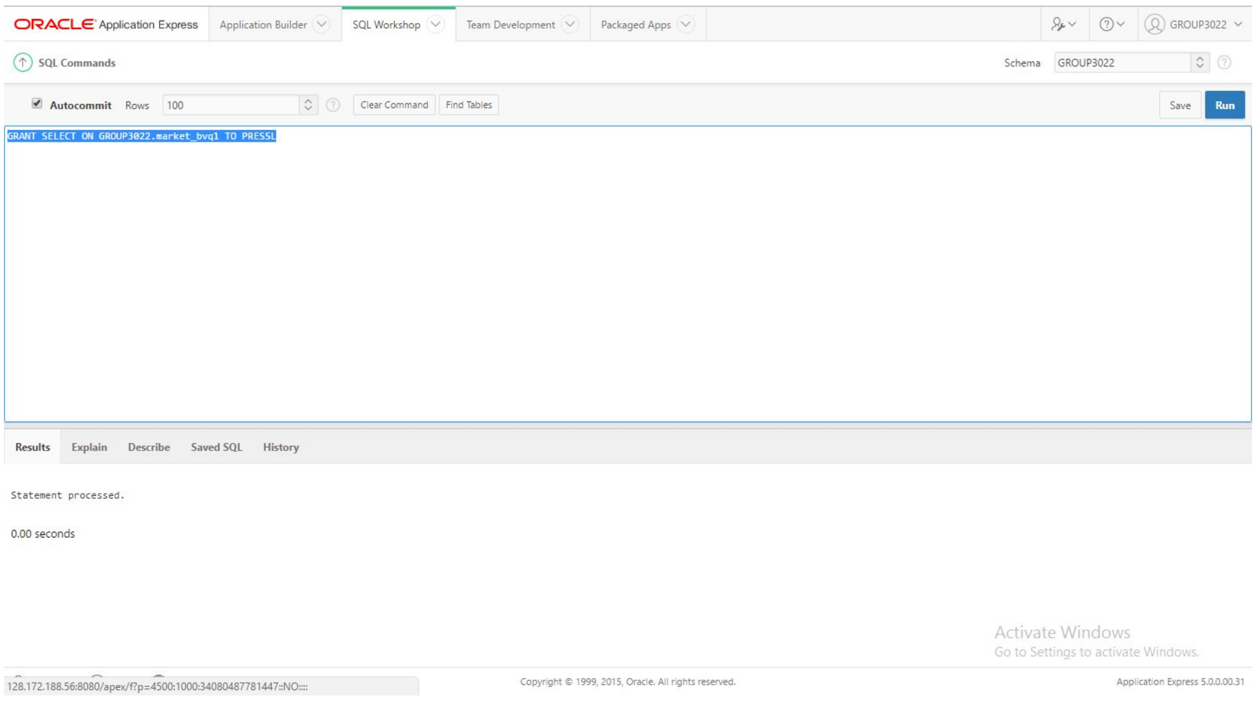
```
GRANT SELECT ON GROUP3022.account_bv1 TO KANGK
```

```
Statement processed.

0.00 seconds
```

Activate Windows
Go to Settings to activate Windows.
group3022
cgu_ist
en
Copyright © 1999, 2015, Oracle. All rights reserved.
Application Express 5.0.0.00.31

Marketing Director

GRANT SELECT ON GROUP3022.market_bvq1 TO PRESSL

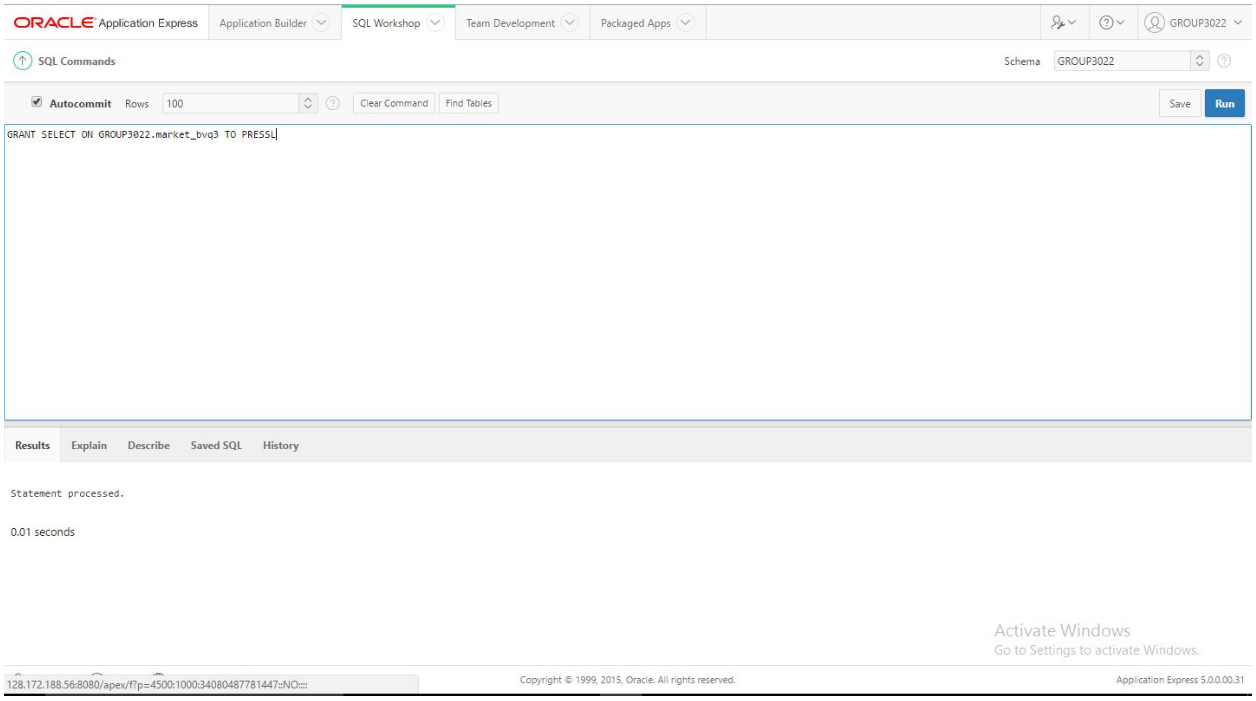

ORACLE Application Express
Application Builder
SQL Workshop
Team Development
Packaged Apps
GROUP3022
SQL Commands
Schema GROUP3022
Autocommit Rows 100
Clear Command
Find Tables
Save
Run
Results
Explain
Describe
Saved SQL
History

```
GRANT SELECT ON GROUP3022.market_bvq1 TO PRESSL
```

```
Statement processed.

0.00 seconds
```

Activate Windows
Go to Settings to activate Windows.
128.172.188.56:8080/apex/f?p=4500:1000:34080487781447::NO:::
Copyright © 1999, 2015, Oracle. All rights reserved.
Application Express 5.0.0.00.31

GRANT SELECT ON GROUP3022.market_bvq3 TO PRESSL



```
GRANT SELECT ON GROUP3022.market_bvq3 TO PRESSL
```

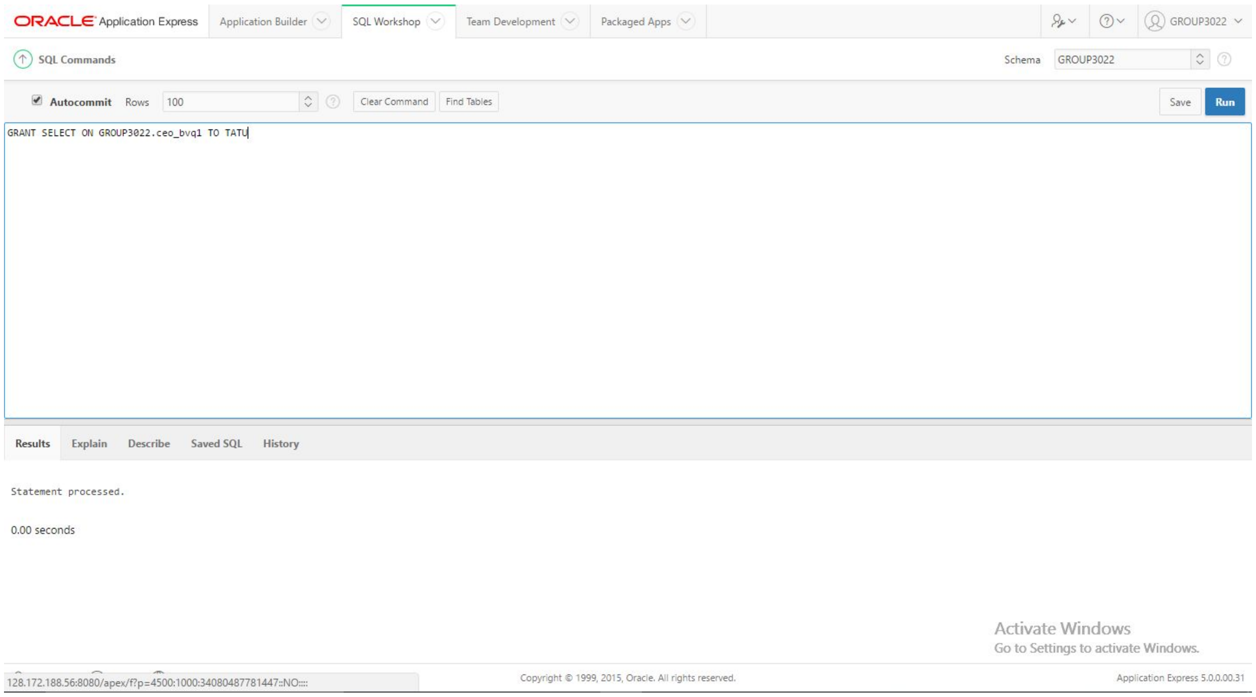
```
Statement processed.

0.01 seconds
```

CEO

Username – TATU

Password - ORAC!E16

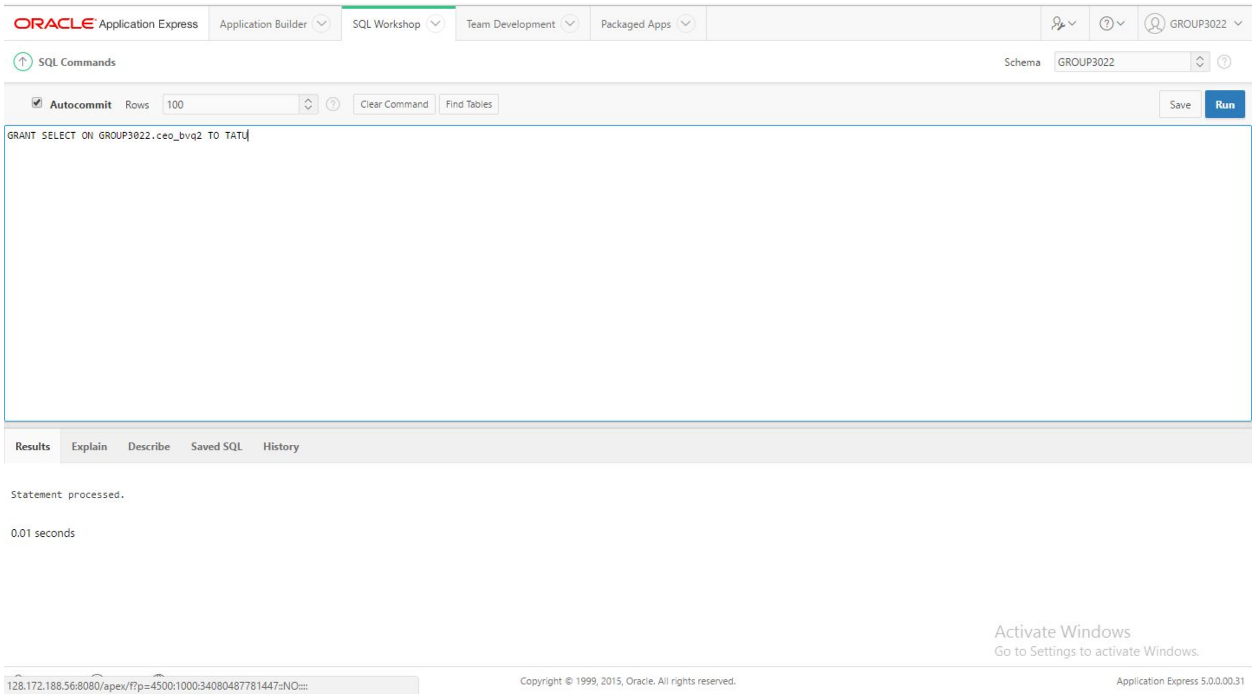
GRANT SELECT ON GROUP3022.ceo_bvq1 TO TATU



```
GRANT SELECT ON GROUP3022.ceo_bvq1 TO TATU
```

```
Statement processed.

0.00 seconds
```

GRANT SELECT ON GROUP3022.ceo_bvq2 TO TATU



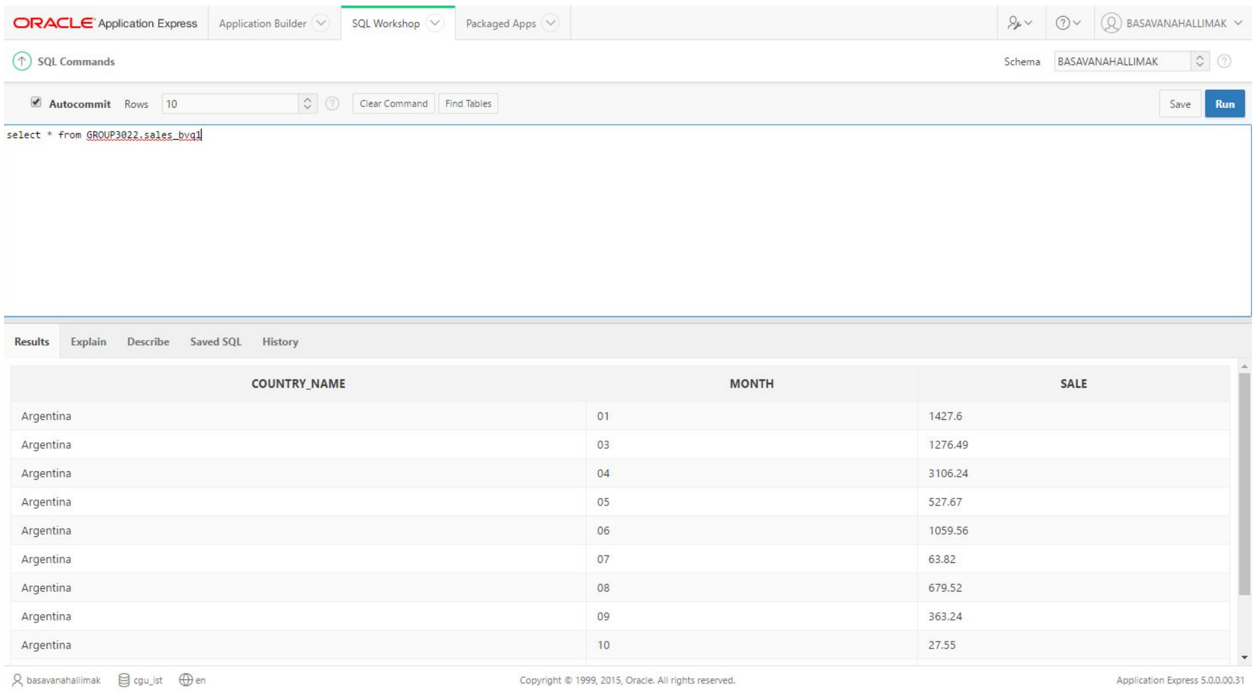
```
GRANT SELECT ON GROUP3022.ceo_bvq2 TO TATU
```

```
Statement processed.

0.01 seconds
```

The following displays positive and negative tests for each user.

Sales Director

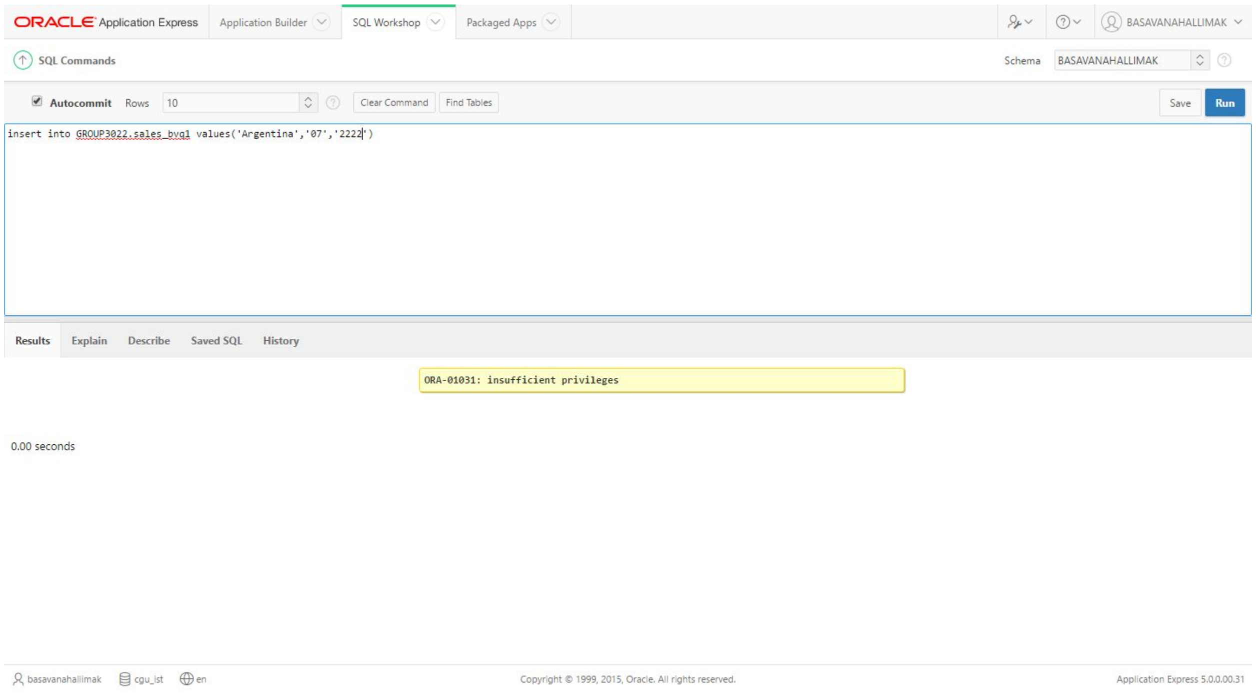
Positive Test



```
select * from GROUP3022.sales_bvq1
```

COUNTRY_NAME	MONTH	SALE
Argentina	01	1427.6
Argentina	03	1276.49
Argentina	04	3106.24
Argentina	05	527.67
Argentina	06	1059.56
Argentina	07	63.82
Argentina	08	679.52
Argentina	09	363.24
Argentina	10	27.55

Negative Test 1

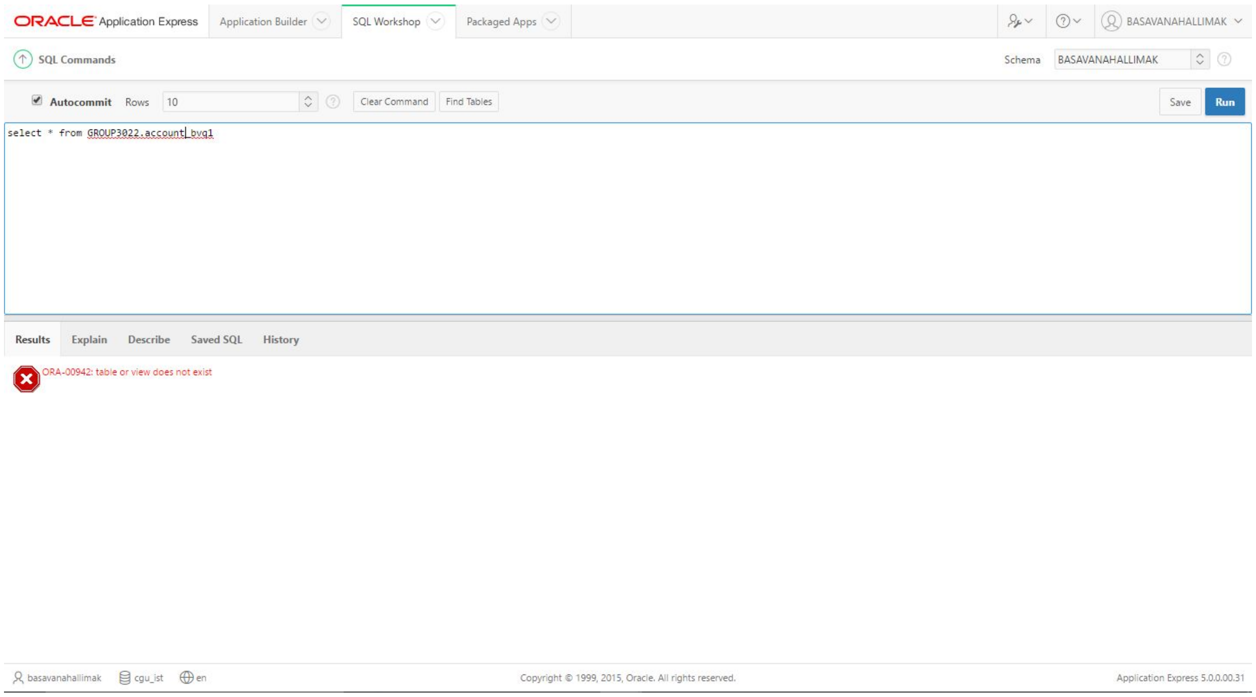


```
insert into GROUP3022.sales_bvq1 values('Argentina','07','2222')
```

```
ORA-01031: insufficient privileges
```

0.00 seconds

Negative Test 2

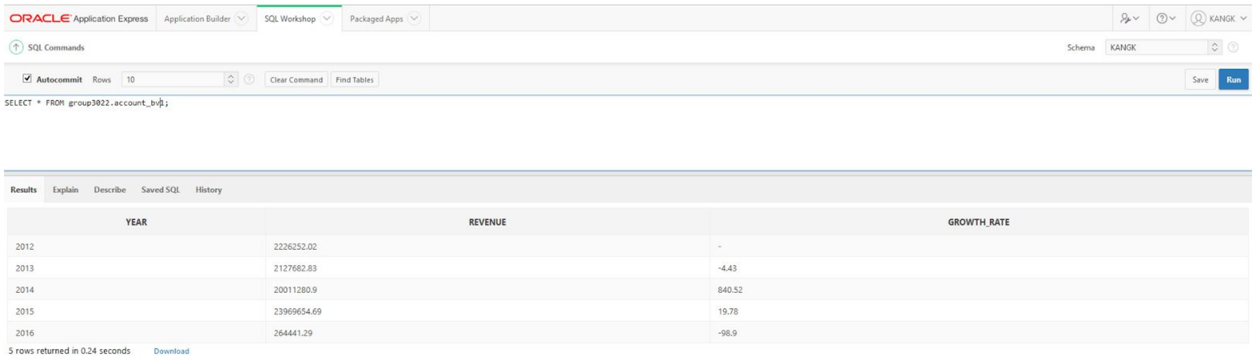


```
select * from GROUP3022.account_bvq1
```

```
ORA-00942: table or view does not exist
```

Accounting Director

Positive Test

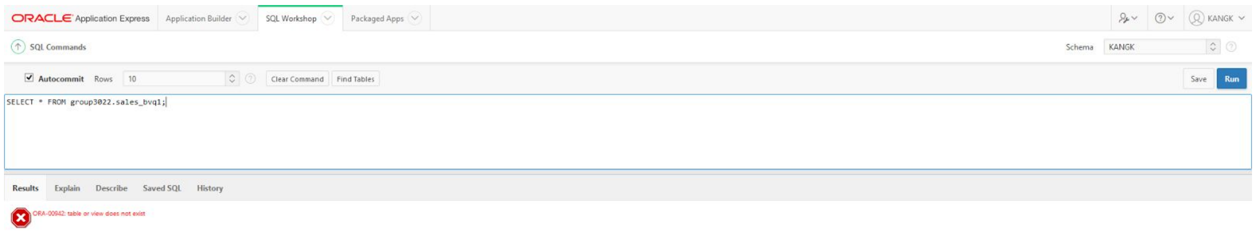


```
SELECT * FROM group3022.account_bv1;
```

YEAR	REVENUE	GROWTH_RATE
2012	2226252.02	-
2013	2127682.83	-4.43
2014	20011280.9	840.52
2015	23969654.69	19.78
2016	264441.29	-98.9

5 rows returned in 0.24 seconds

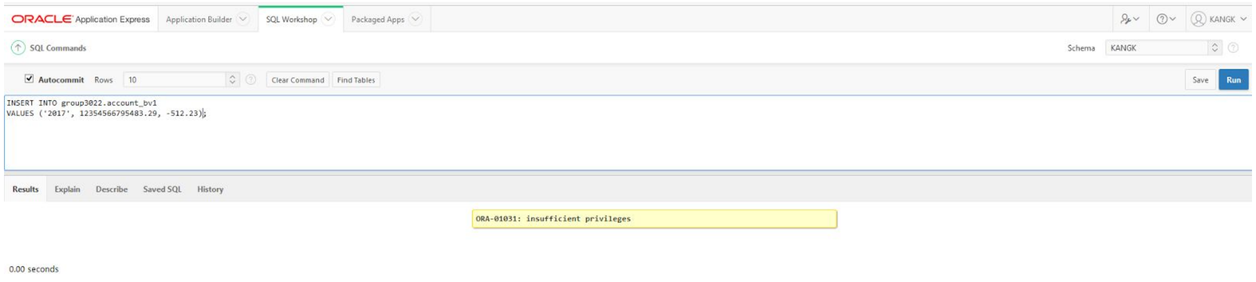
Negative Test 1



```
SELECT * FROM group3022.sales_bvq1;
```

ORA-00942: table or view does not exist

Negative Test 2



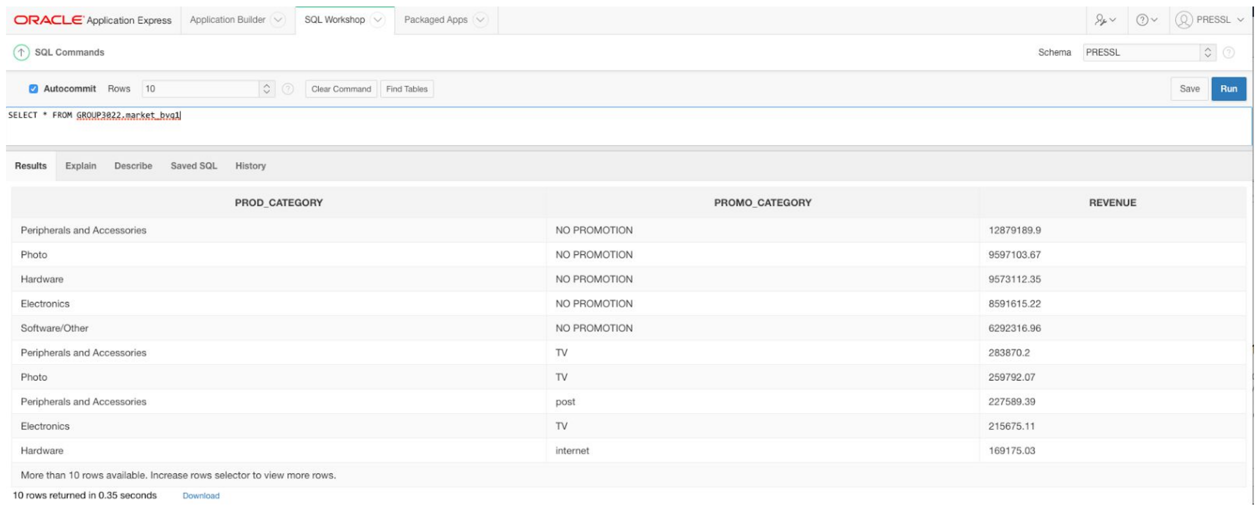
```
INSERT INTO group3022.account_bv1
VALUES ('2017', 12354566795483.29, -512.23);
```

```
ORA-01031: insufficient privileges
```

0.00 seconds

Marketing Director

Positive Test



Schema: PRESSL

Rows: 10

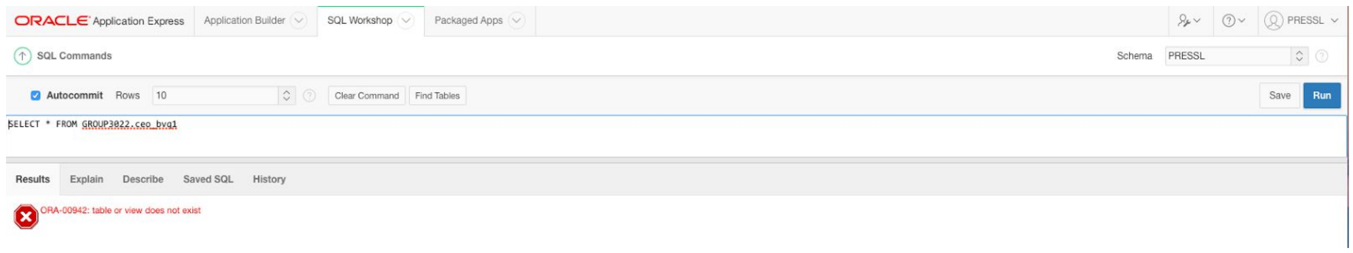
```
SELECT * FROM GROUP3022.market_bvg1
```

PROD_CATEGORY	PROMO_CATEGORY	REVENUE
Peripherals and Accessories	NO PROMOTION	12879189.9
Photo	NO PROMOTION	9597103.67
Hardware	NO PROMOTION	9573112.35
Electronics	NO PROMOTION	8591615.22
Software/Other	NO PROMOTION	6292316.96
Peripherals and Accessories	TV	283870.2
Photo	TV	259792.07
Peripherals and Accessories	post	227589.39
Electronics	TV	215675.11
Hardware	internet	169175.03

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.35 seconds

Negative Test 1



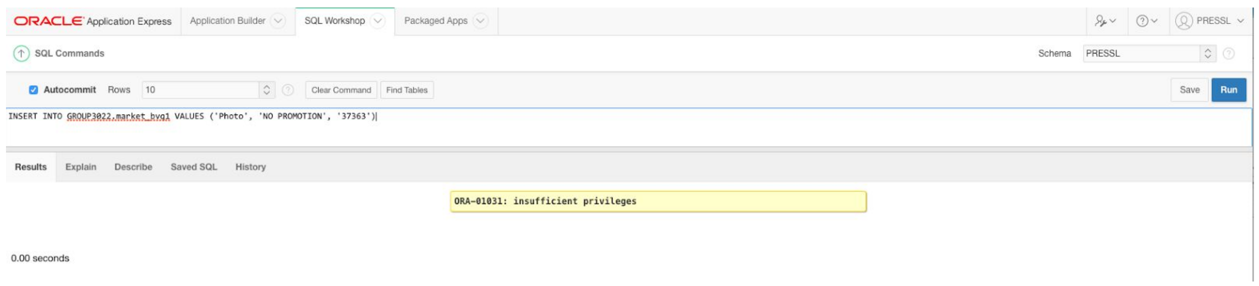
Schema: PRESSL

Rows: 10

```
SELECT * FROM GROUP3022.ceo_bvg1
```

ORA-00942: table or view does not exist

Negative Test 2



Schema: PRESSL

Rows: 10

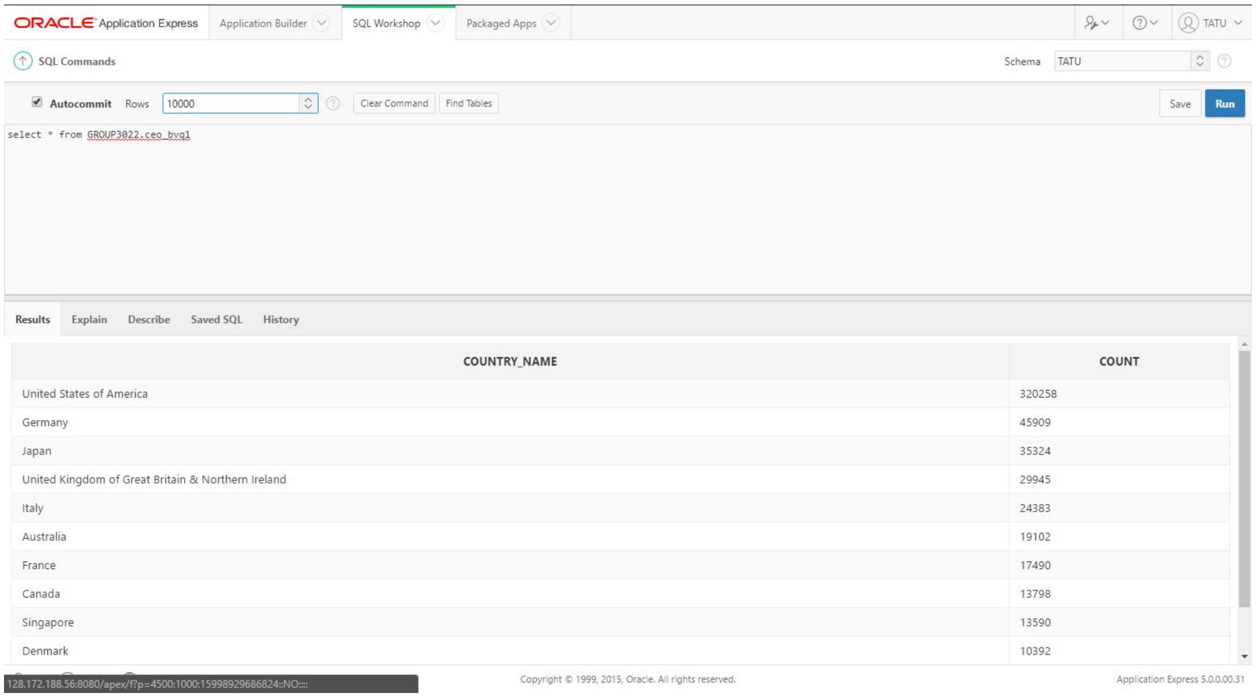
```
INSERT INTO GROUP3022.market_bvg1 VALUES ('Photo', 'NO PROMOTION', '37363')
```

```
ORA-01031: insufficient privileges
```

0.00 seconds

CEO

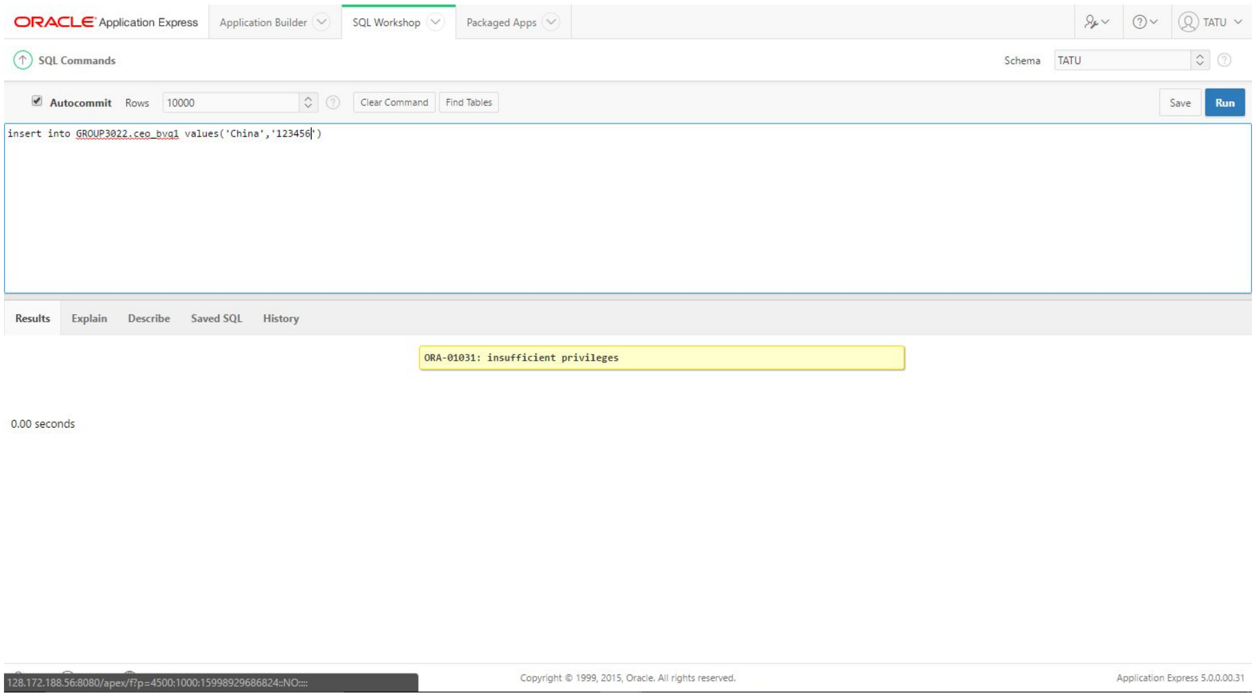
Positive Test



```
select * from GROUP3022.ceo_bvq1
```

COUNTRY_NAME	COUNT
United States of America	320258
Germany	45909
Japan	35324
United Kingdom of Great Britain & Northern Ireland	29945
Italy	24383
Australia	19102
France	17490
Canada	13798
Singapore	13590
Denmark	10392

Negative Test 1

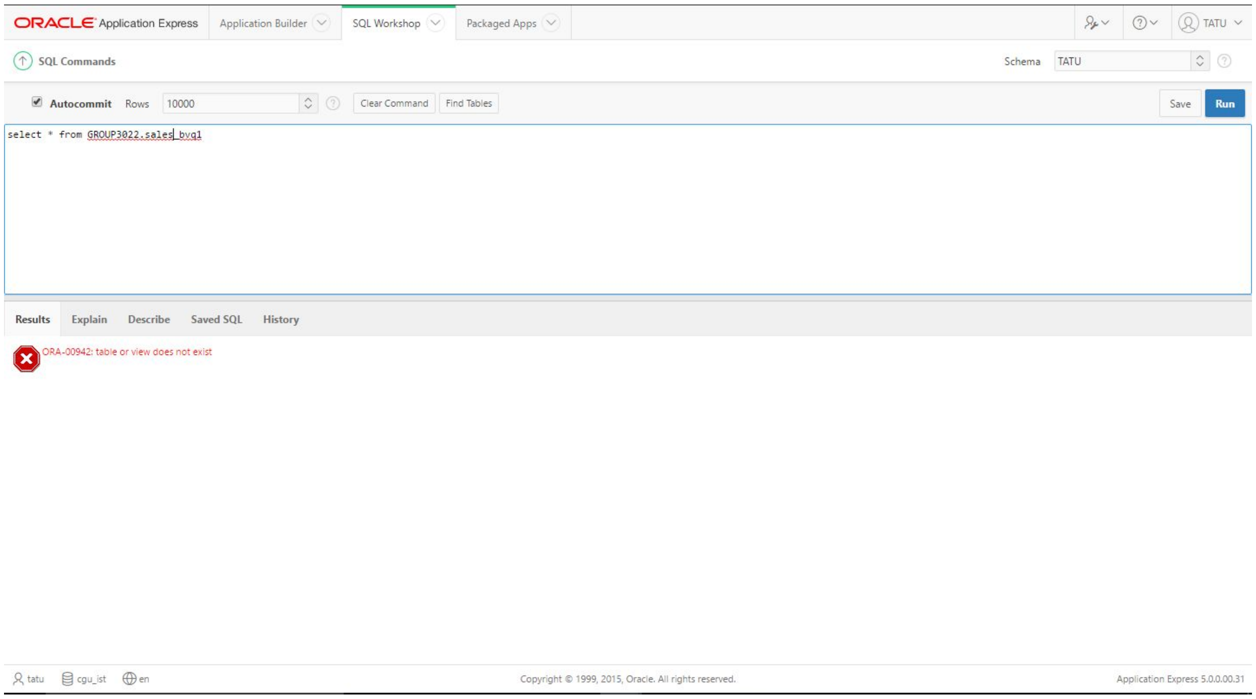


```
insert into GROUP3022.ceo_bvq1 values('China','123456')
```

```
ORA-01031: insufficient privileges
```

0.00 seconds

Negative Test 2



```
select * from GROUP3022.sales_bvq1
```

```
ORA-00942: table or view does not exist
```