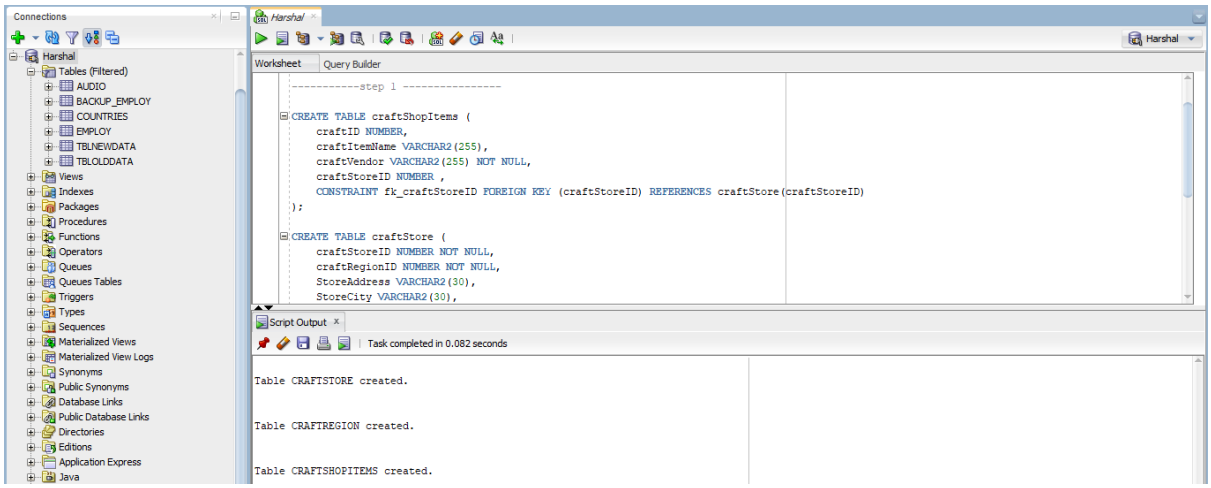
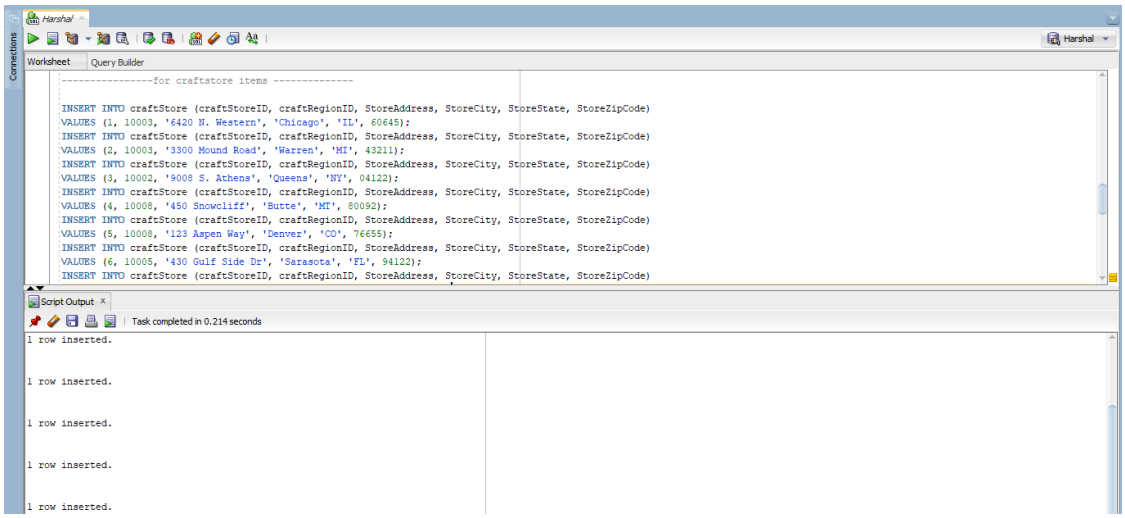
Project one: Creating and Populating a DB Table

Step 1: Creating a Tables



First we needed to create the craftStore and craftRegion tables and then craftShopItems as the craftShopItems had craftStoreID as the foreign key.

Step 2: Populate the tables.



```
select * from craftstore;
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.02 seconds

	CRAFTSTOREID	CRAFTREGIONID	STOREADDRESS	STORECITY	STORESTATE	STOREZIPCODE
1	1	10003	6420 N. Western	Chicago	IL	60645
2	2	10003	3300 Mound Road	Warren	MI	43211
3	3	10002	9008 S. Athens	Queens	NY	4122
4	4	10008	450 Snowcliff	Butte	MT	80092
5	5	10008	123 Aspen Way	Denver	CO	76655
6	6	10005	430 Gulf Side Dr	Sarasota	FL	94122
7	7	10009	1007 Sunset Blvd	Beverly Hills	CA	90210

Saved: Harshal~1

Populating the craftStore

Harshal

Worksheet | Query Builder

```
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10001, 'New England');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10002, 'Mid Atlantic');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10003, 'East North Central');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10004, 'West North Central');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10005, 'South Atlantic');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10006, 'East South Central');
INSERT INTO craftRegion (craftRegionID, craftRegionDescription)
VALUES (10007, 'West South Central');
```

Script Output

Task completed in 0.285 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Script Output x | Query Result x

SQL | All Rows Fetched: 9 in 0.026 seconds

	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	10001	New England
2	10002	Mid Atlantic
3	10003	East North Central
4	10004	West North Central
5	10005	South Atlantic
6	10006	East South Central
7	10007	West South Central
8	10008	Mountain
9	10009	Pacific

Saved: Harshal~1

Populating the craftRegion Table

Harshal

Connections

Worksheet | Query Builder

```
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (1, 'TinkeringToys', 'Toys Deluxe', 2);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (2, 'Dolls', 'Dollie"s', 2);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (3, 'Building Blocks', 'Crafts for Kids', 3);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (4, 'Paint Sets', 'Artiste', 4);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (5, 'Stamps', 'Crafts for Kids', 6);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (6, 'Yarns', 'Yarn Shop', 6);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (7, 'Modeling Clay', 'Toys Deluxe', 1);
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
```

Script Output x

Task completed in 0.243 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Script Output x | Query Result x

SQL | All Rows Fetched: 8 in 0.041 seconds

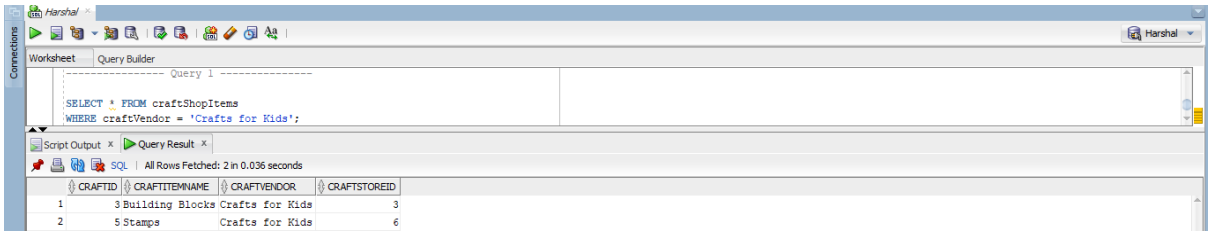
	CRAFTID	CRAFTITEMNAME	CRAFTVENDOR	CRAFTSTOREID
1	1	TinkeringToys	Toys Deluxe	2
2	2	Dolls	Dollie"s	2
3	3	Building Blocks	Crafts for Kids	3
4	4	Paint Sets	Artiste	4
5	5	Stamps	Crafts for Kids	6
6	6	Yarns	Yarn Shop	6
7	7	Modeling Clay	Toys Deluxe	1
8	8	Easels	Artiste	4

Saved: Harshal~1

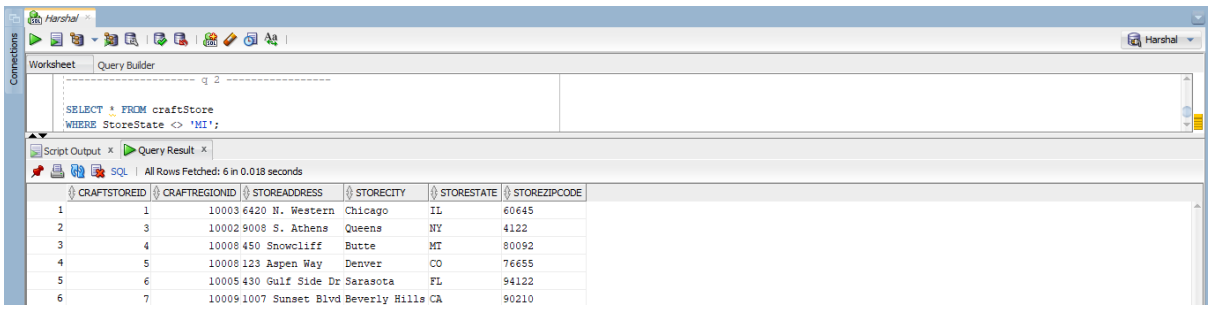
Populating the craftShopItems Table

Step 3 & 4: Performing each queries on the Table and attaching their respective snapshots.

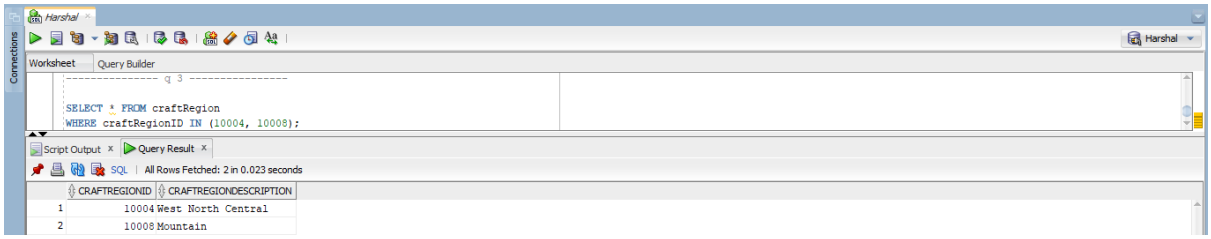
Query 1: Return the records containing Crafts for Kids as craftVendor.



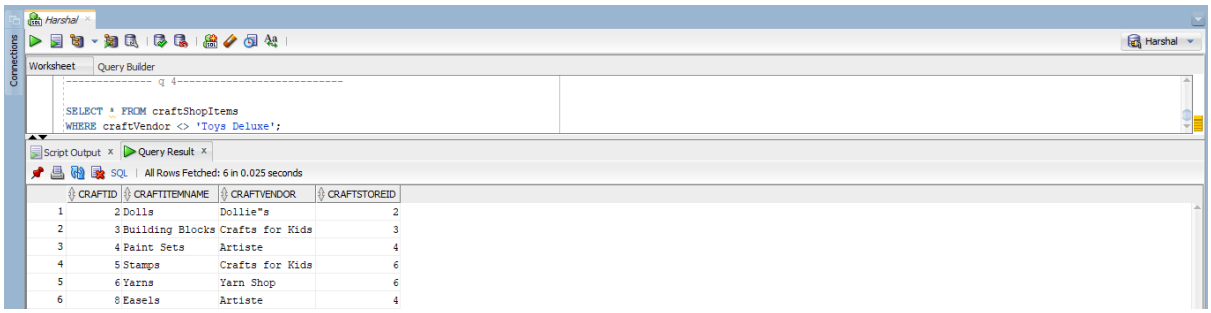
Query 2: Return the records of craftStores that are not located in the state of Michigan.



Query 3: Return the records of craftRegions that are west of the Mississippi river.



Query 4: Return the records of **craftShopItems** that are not vended by Toys Deluxe.

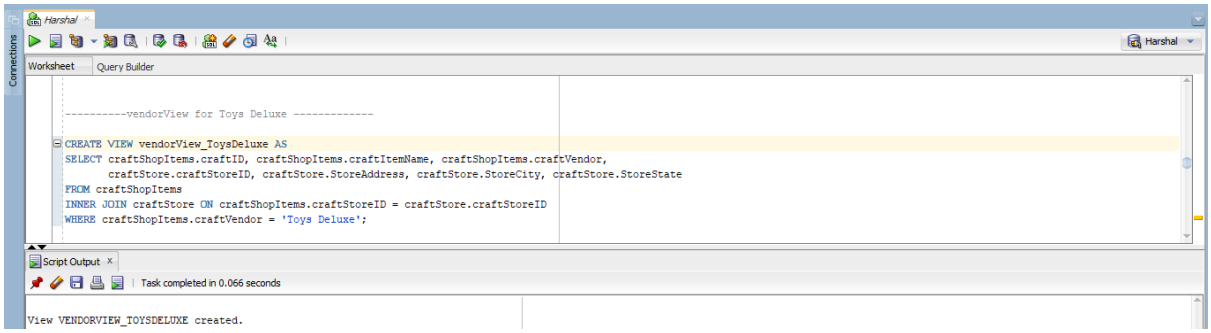


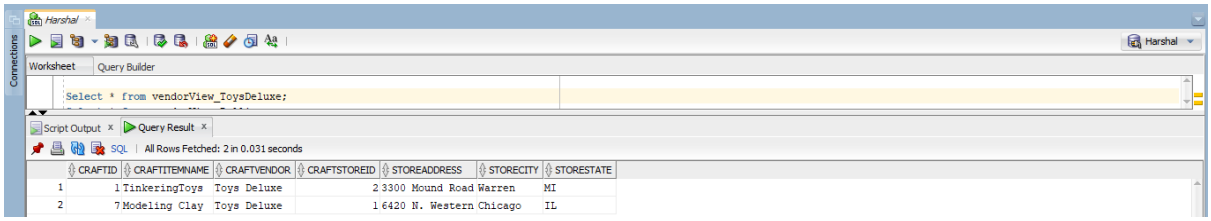
Project 2: Creating the View

Step 1: Create the views. & Step 2 & Step 3: Listing and pasting snapshots

(a)

Toys Deluxe





Dollies

```
---------vendorView for Dollie's ----------------

CREATE VIEW vendorViewDollies AS
SELECT craftShopItems.craftID, craftShopItems.craftItemName, craftShopItems.craftVendor,
       craftStore.craftStoreID, craftStore.StoreAddress, craftStore.StoreCity, craftStore.StoreState
FROM craftShopItems
INNER JOIN craftStore ON craftShopItems.craftStoreID = craftStore.craftStoreID
WHERE craftShopItems.craftVendor = 'Dollie"s';
```

Task completed in 0.083 seconds

```
View VENDORVIEWDOLLIES created.
```

```
Select * from vendorViewDollies;
```

All Rows Fetched: 1 in 0.026 seconds

	CRAFTID	CRAFTITEMNAME	CRAFTVENDOR	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE
1	2	Dolls	Dollie"s	2	3300 Mound Road	Warren	MI

Kids

```
----------vendorView for Crafts for Kids -------------

CREATE VIEW vendorViewCraftForKids AS
SELECT craftShopItems.craftID, craftShopItems.craftItemName, craftShopItems.craftVendor,
       craftStore.craftStoreID, craftStore.StoreAddress, craftStore.StoreCity, craftStore.StoreState
FROM craftShopItems
INNER JOIN craftStore ON craftShopItems.craftStoreID = craftStore.craftStoreID
WHERE craftShopItems.craftVendor = 'Crafts for Kids';
```

Task completed in 0.068 seconds

```
View VENDORVIEWCRAFTFORKIDS created.
```

```
Select * from vendorViewCraftForKids;
```

All Rows Fetched: 2 in 0.024 seconds

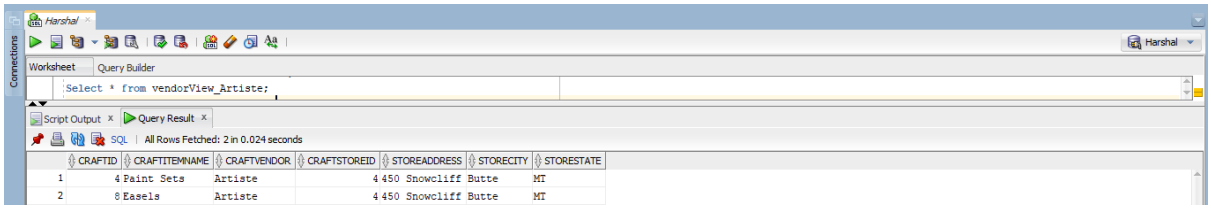
	CRAFTID	CRAFTITEMNAME	CRAFTVENDOR	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE
1	3	Building Blocks	Crafts for Kids	3	9008 S. Athens	Queens	NY
2	5	Stamps	Crafts for Kids	6	430 Gulf Side Dr	Sarasota	FL

Artiste

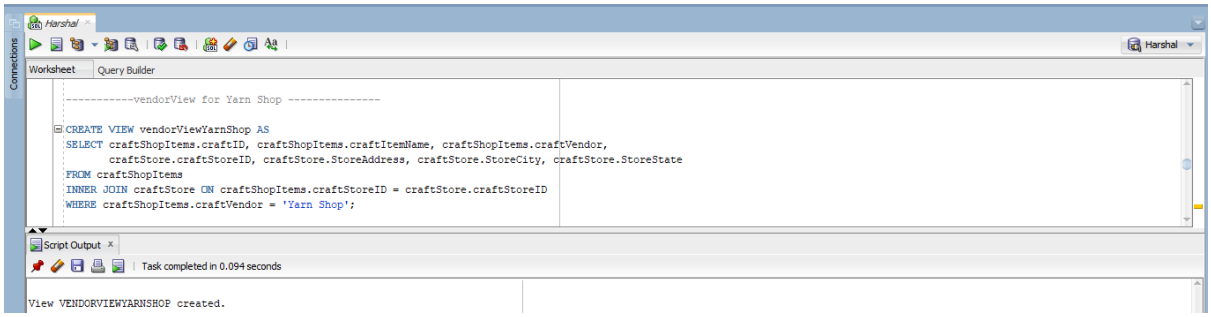
```
-----------vendorView for Artiste ------------

CREATE VIEW vendorView_Artiste AS
SELECT craftShopItems.craftID, craftShopItems.craftItemName, craftShopItems.craftVendor,
       craftStore.craftStoreID, craftStore.StoreAddress, craftStore.StoreCity, craftStore.StoreState
FROM craftShopItems
INNER JOIN craftStore ON craftShopItems.craftStoreID = craftStore.craftStoreID
WHERE craftShopItems.craftVendor = 'Artiste';
```

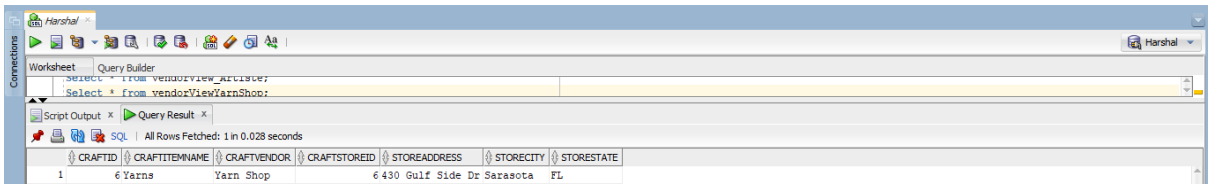
Task completed in 0.092 seconds

```
View VENDORVIEW_ARTISTE created.
```



Yarn

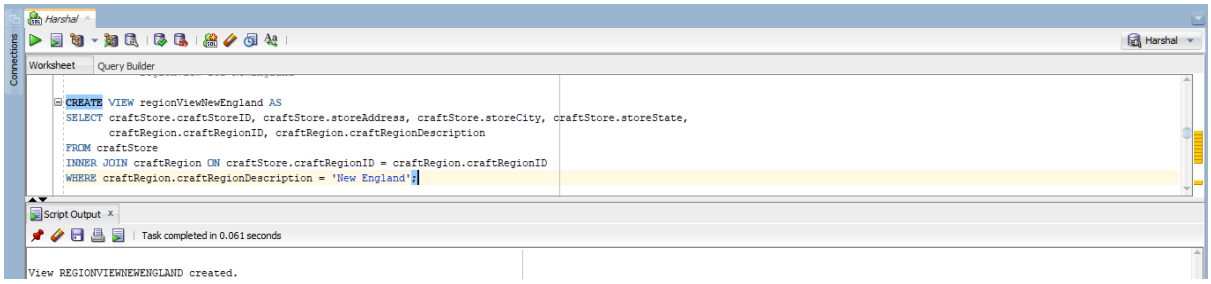


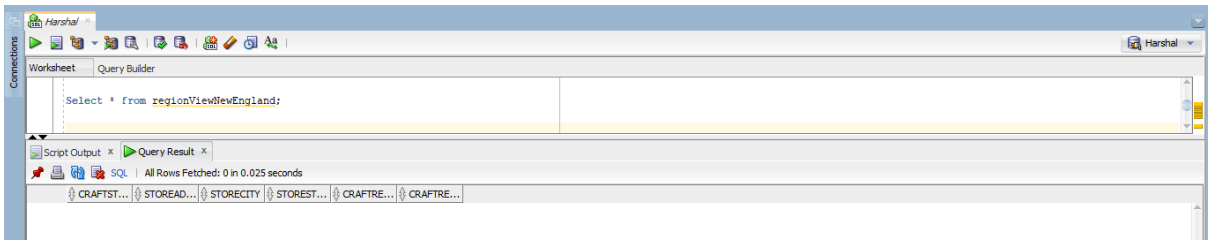


(b)

Regions:

New England





Mid Atlantic

```
CREATE VIEW regionViewMidAtlantic AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'Mid Atlantic';
```

Task completed in 0.08 seconds

```
View REGIONVIEWMIDATLANTIC created.
```

```
Select * from regionViewMidAtlantic;
```

All Rows Fetched: 1 in 0.029 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	3	9008 S. Athens Queens	NY		10002	Mid Atlantic

East North Central

```
------------regionView for East North Central ------------

CREATE VIEW regionView_EastNorthCentral AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'East North Central';
```

Task completed in 0.101 seconds

```
View REGIONVIEW_EASTNORTHCENTRAL created.
```

```
Select * from regionView_EastNorthCentral;
```

All Rows Fetched: 2 in 0.022 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	1	6420 N. Western Chicago	IL		10003	East North Central
2	2	3300 Mound Road Warren	MI		10003	East North Central

West North Central

```
------------regionView for West North Central ------------

CREATE VIEW regionViewWestNorthCentral AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'West North Central';
```

Task completed in 0.066 seconds

```
View REGIONVIEWWESTNORTHCENTRAL created.
```

```
Select * from regionViewWestNorthCentral;
```

All Rows Fetched: 1 in 0.023 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	8	2100 Snowtracks Ln.	Minneapolis	MN	10004	West North Central

South Atlantic

```
----------regionView for SouthAtlantic ----------------

CREATE VIEW regionViewSouthAtlantic AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'South Atlantic';
```

Script Output | Query Result | Task completed in 0.067 seconds

```
View REGIONVIEWSOUTHATLANTIC created.
```

```
Select * from regionViewSouthAtlantic;
```

Script Output | Query Result | All Rows Fetched: 1 in 0.027 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	6	430 Gulf Side Dr	Sarasota	FL	10	005 South Atlantic

East South Central

```
CREATE VIEW regionViewEastSouthCentral AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'East South Central';

Select * from regionViewEastSouthCentral;
```

Script Output | Query Result | Task completed in 0.069 seconds

```
View REGIONVIEWEASTSOUTHCENTRAL created.
```

```
Select * from regionViewEastSouthCentral;
```

Script Output | Query Result | All Rows Fetched: 0 in 0.019 seconds

CRAFTST...	STOREAD...	STORECITY	STOREST...	CRAFTRE...	CRAFTRE...

West South Central

```
----------regionView for WestSouthCentral ----------------

CREATE VIEW regionView_WestSouthCentral AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'West South Central';

Select * from regionView_WestSouthCentral;
```

Script Output | Query Result | Task completed in 0.069 seconds

```
View REGIONVIEWEASTSOUTHCENTRAL created.

View REGIONVIEW_WESTSOUTHCENTRAL created.
```

```
Select * from regionView_WestSouthCentral;
```

All Rows Fetched: 0 in 0.016 seconds

CRAFTST...	STOREAD...	STORECITY	STOREST...	CRAFTRE...	CRAFTRE...

Mountain

```
----------regionView for Mountain ---------------

CREATE VIEW regionView_Mountain AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'Mountain';
```

Task completed in 0.096 seconds

```
View REGIONVIEW_MOUNTAIN created.
```

```
Select * from regionView_Mountain;
```

All Rows Fetched: 2 in 0.016 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	4	450 Snowcliff Butte		MT	10008	Mountain
2	5	123 Aspen Way Denver		CO	10008	Mountain

Pacific

```
----------regionView for Pacific --------------------

CREATE VIEW regionView_Pacific AS
SELECT craftStore.craftStoreID, craftStore.storeAddress, craftStore.storeCity, craftStore.storeState,
       craftRegion.craftRegionID, craftRegion.craftRegionDescription
FROM craftStore
INNER JOIN craftRegion ON craftStore.craftRegionID = craftRegion.craftRegionID
WHERE craftRegion.craftRegionDescription = 'Pacific';
```

Task completed in 0.095 seconds

```
View REGIONVIEW_PACIFIC created.
```

```
Select * from regionView_Pacific;
```

All Rows Fetched: 1 in 0.02 seconds

	CRAFTSTOREID	STOREADDRESS	STORECITY	STORESTATE	CRAFTREGIONID	CRAFTREGIONDESCRIPTION
1	7	1007 Sunset Blvd Beverly Hills		CA	10009	Pacific

Project 3: Creation of Sequence.

Step 1: Create the Sequence

```
CREATE SEQUENCE craftShopItemsSEQ1
    MINVALUE 200
    MAXVALUE 10000
    START WITH 202
    INCREMENT BY 4
    CYCLE;
```

Task completed in 0.072 seconds

```
Sequence CRAFTSHOPITEMSSEQ1 created.
```

Step 2: Use The Sequence

```
INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (craftShopItemsSEQ1.NEXTVAL, 'Colored Chalk', 'DustyProjects', 5);

INSERT INTO craftShopItems (craftID, craftItemName, craftVendor, craftStoreID)
VALUES (craftShopItemsSEQ1.NEXTVAL, 'Craft Wooden Sticks', 'WoodyThings', 7);
```

Task completed in 0.432 seconds

```
1 row inserted.

1 row inserted.
```

```
select * from craftshopitems;
```

All Rows Fetched: 10 in 0.054 seconds

	CRAFTID	CRAFTITEMNAME	CRAFTVENDOR	CRAFTSTOREID
1	1	TinkeringToys	Toys Deluxe	2
2	2	Dolls	Dollie"s	2
3	3	Building Blocks	Crafts for Kids	3
4	4	Paint Sets	Artiste	4
5	5	Stamps	Crafts for Kids	6
6	6	Yarns	Yarn Shop	6
7	7	Modeling Clay	Toys Deluxe	1
8	8	Easels	Artiste	4
9	202	Colored Chalk	DustyProjects	5
10	206	Craft Wooden Sticks	WoodyThings	7

Step 3: Determine the current value of the sequence.

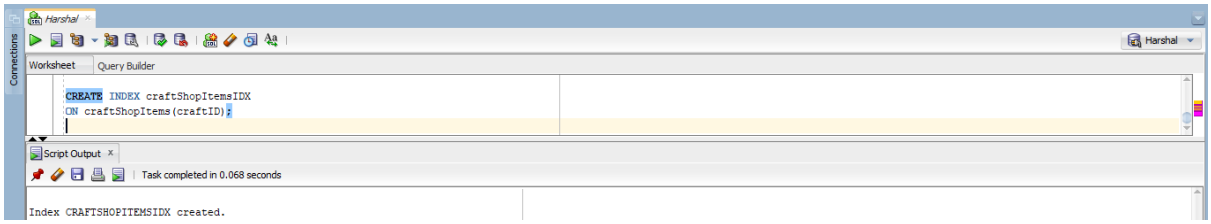
```
---------------- Step 3 ----------------

SELECT craftShopItemsSEQ1.CURRVAL
FROM dual;
```

All Rows Fetched: 1 in 0.043 seconds

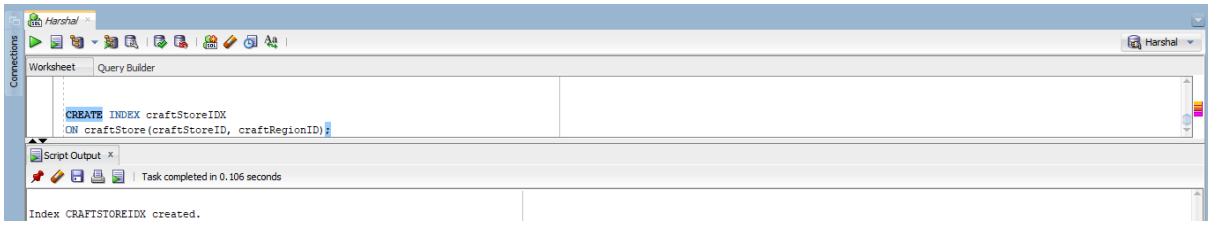
	CURRVAL
1	206

Project 4: Creation of an Index

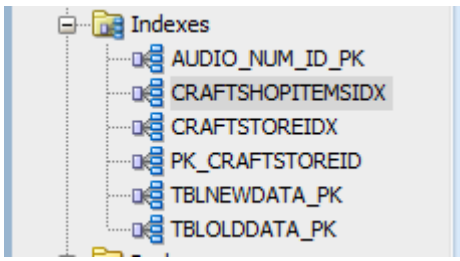
Step 1: Create an Index on a Single Column.



Step 2: Use the Index on Two Columns

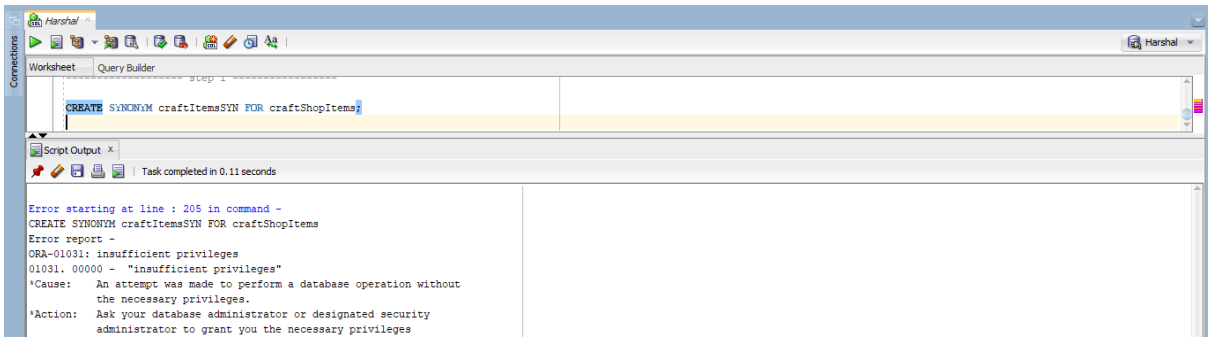


Step 3: Snapshots of all Indexes

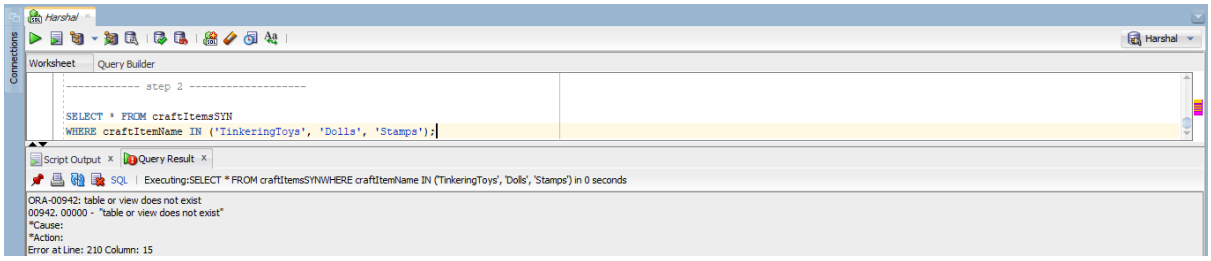


Project 5: Creation of a Synonym

Step 1: Create the Synonym



Step 2: Use the Synonym

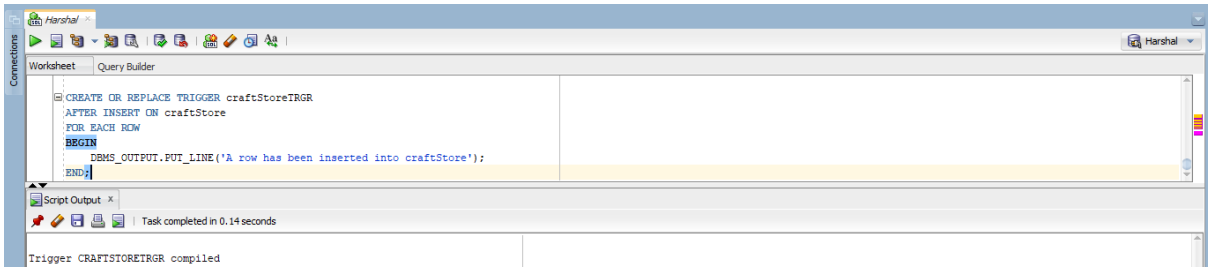


Alternate Query for above:

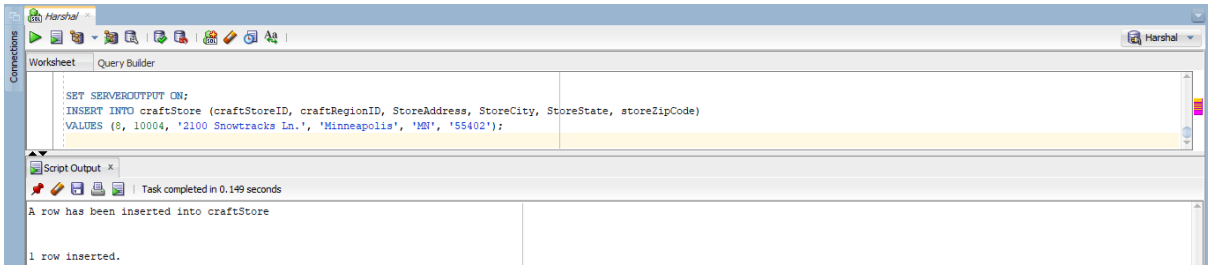
As we don't have the necessary privileges, we cannot create the synonyms. However we can achieve the result by just running the normal select query.

Project 6: Creation of Trigger

Step 1: Create the Trigger



Step 2: Use the Trigger



Project 7: Answer the following questions.

Step 1:

(a)

As a database grows and its tables fill with many entries, adding more database objects such as views, sequences, indexes, and synonyms may be quite beneficial for organizing and retrieving data.

Views: By presenting portions of data from tables, views streamline complicated searches and guarantee consistent data presentation, all while providing streamlined data access. By restricting access to particular data, they improve database security and strengthen security.

Sequences: Sequences provide distinct, increasing values for records by automating the creation of IDs. By preallocating blocks of data, they improve speed by minimizing conflict during ID queries.

Indexes: By establishing an ordered structure for quicker row lookups, indexes speed up data retrieval. Indexes minimize the amount of data scanned in order to maximize query performance as data volume grows.

Synonyms: Synonyms improve code readability and streamline searches by offering user-friendly, alternate names for database objects. They disentangle dependencies so that modifications to the database schema may be made without impacting scripts or programs that use synonyms.

(b)

Database profiles are essential for resource allocation, user access management, and security measure enforcement. For database users, they should be made for a number of reasons:

Resource Management: Profiles assist in allocating CPU, memory, and I/O limits—as well as other database resources—for various user groups or individuals. By doing this, it is impossible for one user to monopolize system resources, guaranteeing equitable distribution and peak performance for everyone.

Security and Access Control: By imposing restrictions on user sessions, they enable administrators to implement security rules. This involves preventing unwanted operations from occurring and restricting access to particular tables, schemas, or functionality in order to improve overall database security. Control and Customization: Profiles allow experiences for

users to be tailored to their requirements or responsibilities. They can, for instance, impose session timeouts, password rules, or settings unique to certain user groups.

Performance Optimization: Database administrators (DBAs) may fine-tune characteristics such as query execution time, parallelism, or resource consumption thresholds for certain user types by specifying particular parameters in profiles.

Users are allocated profiles according to their jobs, responsibilities, and access needs. They maintain an organized and regulated environment inside the database, adjusting performance, security, and resource use to meet the demands of different user groups or people.

(c)

A view's capacity to update a table is dependent on a number of variables, mostly the view's design.

Updatable Views: If a view is deemed updatable, then newly added records via it have the ability to change the underlying table. A view that permits changes (insert, update, and delete) to the underlying table or tables via the view is called an updatable view. Not every viewpoint, though, can be updated automatically.

A view needs to fulfill certain requirements in order to be updatable:

Only one base table may be referenced by the view.

The columns in the base table that are being changed in the view must match those columns exactly.

Certain query constructs, like as aggregate functions, GROUP BY, DISTINCT, etc., must not be included in the view.

Data Transformation Statements with Views: The SQL standard states that there is a cap on the quantity of data manipulation statements that may be used in conjunction with views. In general, INSERT, UPDATE, and DELETE statements can be used to alter a single table view (simple view) that satisfies the requirements for updatability. However, the ability to change data through the view may be limited or prohibited when working with more complicated views or those that incorporate many tables. *(Ma & Wang, 2011)*

(d)

By explicitly designating the value to be used for the subsequent sequence value or by changing the sequence itself, a Database Administrator (DBA) can override the record value created by a sequence. This may be done by changing the sequence to a specified value or by modifying the current value of the sequence using SQL instructions.

An example for that can be:

```
CREATE SEQUENCE sequenceXYZ
    START WITH 1
```

INCREMENT BY 1;

ALTER SEQUENCE sequenceXYZ RESTART WITH 100;

This command overrides the sequence's default incrementation and resets it to start producing values at 100.

Another way to achieve the same can be using the data insertion using the basic normal query.

(e)

Triggers serve different roles in database architecture depending on whether they are triggered before or after Data Manipulation Language (DML) activities.

BEFORE Triggers: Perfect for applying restrictions or verifying data before DML operations. They alter incoming data, stop illegal activity, or make ensuring prerequisites are satisfied before authorizing an operation.

AFTER Triggers: Suitable for post-DML operations such as updating derived data, preserving referential integrity, or recording modifications following the conclusion of the DML activity. beneficial for establishing uniformity between tables or conducting audits.

Performance effects are taken into account; response times may be impacted by pre-operation validations BEFORE triggers, whilst data consistency is ensured post-action AFTER triggers.

Depending on your needs, you may choose between BEFORE and AFTER triggers to manage activities post-operation, such as reporting changes or preserving data integrity, or to handle pre-action tasks like data transformation or validation.

Step 2: Further considerations for this Database Project

(a)

In order to handle issues that may arise during trigger execution, exception handling in triggers is essential. Typical exception handling methods used in triggers include the following:

Exception logging is the process of identifying exceptions inside of triggers and recording specific error messages or other relevant data into a log table or error handling system. This makes mistake analysis and troubleshooting easier.

Error Notification: Notifying administrators or other pertinent parties when exceptions arise from triggers. This guarantees that problems that might impair database operations receive quick attention.

Rollback Transactions: To preserve data consistency and integrity, handle exceptions by rolling back the transaction as necessary. This stops updates from being made that are inaccurate or incomplete.

Custom Error Handling: Applying certain procedures according to the kind of error that has been detected. To minimize the problem, one might attempt retrying the process, implementing alternate logic, or taking fallback actions. SQLERRM can be used to print the Exception using DBMS_OUTPUT.PUT_LINE().

User-Friendly Messages: Giving users or programs interacting with the database concise, understandable error messages that help them resolve problems brought on by exceptions connected to triggers.

Trigger exception management basically entails identifying mistakes, recording pertinent data, alerting stakeholders, preserving data consistency, and providing advice or other solutions to address problems that arise during trigger execution.

(b)

The "craftShopItems" table, which has columns for the craft ID, item name, seller, and store ID, identifies particular products in a craft shop. Similar information is contained in the "craftStore" database, which also provides shop and region IDs, locations, and cities for craft retailers. Furthermore, areas associated to crafts are defined by region IDs and descriptions in the "craftRegion" database.

Although the layout of these tables is conventional, a more thorough study necessitates knowing the details of each criteria and how they relate to one another. If linkages or dependencies between the tables aren't totally obvious from the information presented, then more standardization could be required. So, basically, we don’t require any more normalization here at moment.

(c)

The "craftID" field functions as the primary key in the "craftShopItems" database, automatically generating a unique index for effective identification and guaranteeing uniqueness within that column.

Similar to this, the "craftStoreID" serves as the primary key in the "craftStore" table, creating a distinct index that allows each row in the database to be uniquely identified.

Like in the other tables, the "craftRegionID" serves as the primary key in the "craftRegion" table, automatically generating a unique index.

When a primary key is defined on a column in a relational database, the column's unique index is automatically created, guaranteeing that every value is distinct and promoting quick data retrieval.

(d)

Yes, we can observe the time taken to execute the queries in the Oracle SQL Developer. The snapshot for the same is attached below.

```
Task completed in 0.149 seconds
```

This was the time taken to execute a basic select query which is 149 milliseconds.