Sure, let's start with Materialized Views.

Materialized Views:

1. **Creating Materialized Views:**

   A materialized view is a database object that contains the results of a query. Unlike regular views, a materialized view stores the actual data, making it suitable for improving query performance.

   Example 1: Using PostgreSQL

   ```sql
   CREATE MATERIALIZED VIEW mv_example AS
   SELECT column1, column2, SUM(column3) AS total
   FROM table1
   GROUP BY column1, column2;
   ```

   In this example, we create a materialized view named `mv_example` that aggregates data from `table1` based on columns `column1` and `column2`, calculating the total sum of `column3`.

   Example 2: Using Oracle

   ```sql
   CREATE MATERIALIZED VIEW mv_example
   BUILD IMMEDIATE
   REFRESH COMPLETE
   AS
   SELECT column1, column2, SUM(column3) AS total
   FROM table1
   GROUP BY column1, column2;
   ```

   Here, we create a materialized view with immediate build and complete refresh options in Oracle. The `BUILD IMMEDIATE` clause indicates that the materialized view should be populated immediately, and `REFRESH COMPLETE` specifies that a complete refresh should be performed.

2. **Refreshing Materialized Views:**

   Materialized views need to be refreshed to update their data based on changes in the underlying tables.

Example 1: Using PostgreSQL

```sql
REFRESH MATERIALIZED VIEW mv_example;
```

In PostgreSQL, you can manually refresh a materialized view using the `REFRESH` command.

Example 2: Using Oracle

```sql
DBMS_MVIEW.REFRESH('MV_EXAMPLE', 'C');
```

In Oracle, you can use the `DBMS_MVIEW` package to refresh a materialized view. The `'C'` parameter indicates a complete refresh.

Now, let's move on to Database Migration.

Database Migration:

Database migration involves moving from one database system to another or upgrading to a new version. Tools and techniques vary depending on the specific migration scenario.

1. **Tools and Techniques for Migrating Databases:**

   - **Example 1: Using the AWS Database Migration Service (DMS):**

     AWS DMS is a managed service that supports various database migration scenarios.

     ```json
     {
       "Type": "AWS::DMS::Endpoint",
       "Properties": {
         "EndpointIdentifier": "source-endpoint",
         "EndpointType": "source",
         "EngineName": "mysql",
         "Username": "source_username",
         "Password": "source_password",
         "ServerName": "source_endpoint",
         "Port": 3306
       }
```

```
}
```

This CloudFormation template snippet creates a source endpoint for migrating data from a MySQL database.

- **Example 2: Using the Django ORM for Schema Migration:**

  Django provides a powerful Object-Relational Mapping (ORM) for Python applications.

  ```python
  # Create a migration file
  python manage.py makemigrations

  # Apply the migration
  python manage.py migrate
  ```

  Django's ORM allows you to define database schemas in Python code. The `makemigrations` command generates migration files, and `migrate` applies those changes to the database.

Remember that specific tools and techniques may vary based on the databases involved, the scale of the migration, and other factors. Always refer to the documentation of the tools and databases in use for the most accurate and up-to-date information.

Certainly, let's delve deeper into the topic.

Materialized Views:

3. **Creating Materialized Views in Oracle with Joins:**

   Materialized views can involve joins for more complex aggregations.

   Example 3: Using Oracle with Joins

   ```sql
   CREATE MATERIALIZED VIEW mv_complex_example
   BUILD IMMEDIATE
   REFRESH FAST
   AS
   SELECT t1.column1, t1.column2, t2.column4
   FROM table1 t1
   JOIN table2 t2 ON t1.column3 = t2.column3;
```

```

This Oracle example demonstrates creating a materialized view that includes a join between `table1` and `table2` based on the equality of `column3`.

4. **Refreshing Materialized Views with Incremental Refresh:**

Incremental refresh is a technique where only the changed data is updated in the materialized view.

**Example 4: Using PostgreSQL with Incremental Refresh**

```sql
REFRESH MATERIALIZED VIEW mv_incremental_refresh WITH DATA;
```

In PostgreSQL, the `WITH DATA` clause during refresh ensures that only the changed data is updated in the materialized view.

Database Migration:

3. **Using the Flask-Migrate Extension for SQLAlchemy:**

Flask-Migrate is an extension for Flask applications that integrates with SQLAlchemy for database migrations.

Example 3: Using Flask-Migrate

```python
# Create a migration
flask db migrate -m "Add new_table"

# Apply the migration
flask db upgrade
```

This Flask example demonstrates creating and applying a migration to add a new table to the database using Flask-Migrate.

4. **Cross-Database Migration with Flyway:**

Flyway is a database migration tool that supports cross-database migrations.

Example 4: Using Flyway
```

```
# Create a SQL migration script
V1__create_table.sql

-- SQL statements for creating a table

# Apply the migration
flyway migrate
```

Flyway allows you to define database changes using SQL migration scripts. The `flyway migrate` command applies these scripts, making it suitable for cross-database migrations.

These examples provide a more comprehensive view of creating materialized views with joins and handling incremental refreshes. Additionally, they showcase Flask-Migrate for Python-based migrations and Flyway for a cross-database migration scenario. Always refer to the documentation of the specific tools and databases for detailed information and best practices.