**//STEP 1. open and change file: \vendor\magento\module-url-rewrite\Model\Storage\DbStorage.php  
  
//CHANGE WHOLE FUNCTION FROM:**  
protected function doReplace($urls)  
 {  
 foreach ($this->createFilterDataBasedOnUrls($urls) as $type => $urlData) {  
 $urlData[UrlRewrite::ENTITY\_TYPE] = $type;  
 $this->deleteByData($urlData);  
 }  
 $data = [];  
 foreach ($urls as $url) {  
 $data[] = $url->toArray();  
 }  
  
 foreach ($data as $key => $info) {  
 if (isset($info['target\_path']) && stristr($info['target\_path'], '/category/1') && $info['entity\_type'] == 'product') {  
 unset($data[$key]);  
 }  
  
 $this->insertMultiple($data);  
 }  
 }  
   
**//TO THIS ONE:**  
protected function doReplace($urls)  
{  
 foreach ($this->createFilterDataBasedOnUrls($urls) as $type => $urlData) {  
 $urlData[UrlRewrite::ENTITY\_TYPE] = $type;  
 $this->deleteByData($urlData);  
 }  
 $data = [];  
 $storeId\_requestPaths = [];  
 foreach ($urls as $url) {  
 $storeId = $url->getStoreId();  
 $requestPath = $url->getRequestPath();  
 // Skip if is exist in the database  
 $sql = "SELECT \* FROM url\_rewrite where store\_id = $storeId and request\_path = '$requestPath'";  
 $exists = $this->connection->fetchOne($sql);  
  
 if ($exists) continue;  
  
 $storeId\_requestPaths[] = $storeId . '-' . $requestPath;  
 $data[] = $url->toArray();  
 }  
  
 // Remove duplication data;  
 $n = count($storeId\_requestPaths);  
 for ($i = 0; $i < $n - 1; $i++) {  
 for ($j = $i + 1; $j < $n; $j++) {  
 if ($storeId\_requestPaths[$i] == $storeId\_requestPaths[$j]) {  
 unset($data[$j]);  
 }  
 }  
 }  
 $this->insertMultiple($data);  
}  
  
  
**//STEP 2. open and change file: \vendor\magento\module-catalog-url-rewrite/Model/Product/AnchorUrlRewriteGenerator.php (L58)  
//whole function from:**  
public function generate($storeId, Product $product, ObjectRegistry $productCategories)  
 {  
 $urls = [];  
 foreach ($productCategories->getList() as $category) {  
 $anchorCategoryIds = $category->getAnchorsAbove();  
 if ($anchorCategoryIds) {  
 foreach ($anchorCategoryIds as $anchorCategoryId) {  
 $anchorCategory = $this->categoryRepository->get($anchorCategoryId);  
 $urls[] = $this->urlRewriteFactory->create()  
 ->setEntityType(ProductUrlRewriteGenerator::ENTITY\_TYPE)  
 ->setEntityId($product->getId())  
 ->setRequestPath(  
 $this->urlPathGenerator->getUrlPathWithSuffix(  
 $product,  
 $storeId,  
 $anchorCategory  
 )  
 )  
 ->setTargetPath(  
 $this->urlPathGenerator->getCanonicalUrlPath(  
 $product,  
 $anchorCategory  
 )  
 )  
 ->setStoreId($storeId)  
 ->setMetadata(['category\_id' => $anchorCategory->getId()]);  
 }  
 }  
 }  
  
 return $urls;  
 }  
  
**//TO THIS :**  
  
 public function generate($storeId, Product $product, ObjectRegistry $productCategories)  
 {  
 $urls = [];  
 foreach ($productCategories->getList() as $category) {  
 $anchorCategoryIds = $category->getAnchorsAbove();  
 if ($anchorCategoryIds) {  
 foreach ($anchorCategoryIds as $anchorCategoryId) {  
 //Default: $anchorCategory = $this->categoryRepository->get($anchorCategoryId);  
 $anchorCategory = $this->categoryRepository->get($anchorCategoryId, $storeId);  
 $urls[] = $this->urlRewriteFactory->create()  
 ->setEntityType(ProductUrlRewriteGenerator::ENTITY\_TYPE)  
 ->setEntityId($product->getId())  
 ->setRequestPath(  
 $this->urlPathGenerator->getUrlPathWithSuffix(  
 $product,  
 $storeId,  
 $anchorCategory  
 )  
 )  
 ->setTargetPath(  
 $this->urlPathGenerator->getCanonicalUrlPath(  
 $product,  
 $anchorCategory  
 )  
 )  
 ->setStoreId($storeId)  
 ->setMetadata(['category\_id' => $anchorCategory->getId()]);  
 }  
 }  
 }  
  
 return $urls;  
 }  
  
  
**//STEM 3. open and change file: \vendor\magento\module-catalog-url-rewrite/Model/ProductUrlRewriteGenerator.php (L146)  
  
//whole function from this :**  
  
protected function generateForGlobalScope($productCategories)  
 {  
 $urls = [];  
 $productId = $this->product->getEntityId();  
 foreach ($this->product->getStoreIds() as $id) {  
 if (!$this->isGlobalScope($id)  
 && !$this->storeViewService->doesEntityHaveOverriddenUrlKeyForStore($id, $productId, Product::ENTITY)  
 ) {  
 $urls = array\_merge($urls, $this->generateForSpecificStoreView($id, $productCategories));  
 }  
 }  
 return $urls;  
 }  
  
//TO THIS :   
  
protected function generateForGlobalScope($productCategories)  
 {  
 $urls = [];  
 $productId = $this->product->getEntityId();  
 foreach ($this->product->getStoreIds() as $id) {  
 if (!$this->isGlobalScope($id)  
 && !$this->storeViewService->doesEntityHaveOverriddenUrlKeyForStore($id, $productId, Product::ENTITY)  
 ) {  
 // Default: $urls = array\_merge($urls, $this->generateForSpecificStoreView($id, $productCategories));  
   
 // before loading the category collection by looping it, clone it and set the correct store id,  
 // so we get the correct url\_path & url\_key for that specific store id  
 $storeSpecificProductCategories = clone $productCategories;  
 $storeSpecificProductCategories->setStoreId($id);  
  
 $urls = array\_merge($urls, $this->generateForSpecificStoreView($id, $storeSpecificProductCategories));  
 }  
 }  
 return $urls;  
 }  
   
**//THE END - NOW TEST :)**  
   
   
**//THATS ALL   
// ALL CHANGES READ MORE IN DESCRIPTION   
//IF THIS WAS HELPFULL PRESS +1 - THANKS !   
  
:)**